



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1971

An implemented transformational scheme for monic polynomials with integer coefficients

Lutz, Robert Edward, Jr.

Monterey, California. Naval Postgraduate School

<http://hdl.handle.net/10945/15753>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

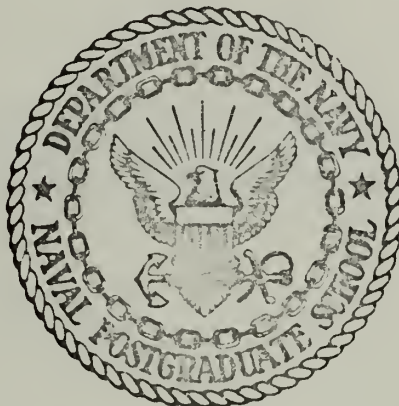
Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

AN IMPLEMENTED TRANSFORMATIONAL
SCHEME FOR MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS

Robert Edward Lutz, Jr.

United States Naval Postgraduate School



THESIS

AN IMPLEMENTED TRANSFORMATIONAL SCHEME
FOR
MONIC POLYNOMIALS WITH INTEGER COEFFICIENTS

by

Robert Edward Lutz, Jr.

Thesis Advisor:

Daniel L. Davis

June 1971

Approved for public release; distribution unlimited.

T139921



An Implemented Transformational Scheme
for
Monic Polynomials with Integer Coefficients

by

Robert Edward Lutz, Jr.
Lieutenant, United States Navy
B.S., Auburn University, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1971

ABSTRACT

A discussion of the problem of the irreducibility of polynomials in the ring of integral polynomials establishes the framework of the research. A transformational scheme is postulated to facilitate investigation of the problem. The coherency of the scheme is detailed and the necessary computational techniques developed.

To determine the efficacy of the transformational scheme, the specification and collection of appropriate sets of data are discussed. The transformational scheme is then applied to the data and the results are tabulated and discussed.

The conclusion is drawn that a transformational scheme is a useful tool for the investigation of the irreducibility of this type of polynomial and some hypotheses as to the refinement and extension of the technique are stated.

TABLE OF CONTENTS

I.	INTRODUCTION -----	5
II.	TRANSFORMATIONAL SCHEME -----	7
III.	DEVELOPMENT OF ALGORITHMS -----	15
IV.	SPECIFICATION AND COLLECTION OF DATA -----	25
V.	APPLICATION OF TRANSFORMATIONAL SCHEME -----	34
VI.	RESULTS, CONCLUSIONS, AND HYPOTHESES -----	36
APPENDIX A:	TABLES OF IRREDUCIBLE POLYNOMIALS -----	39
APPENDIX B:	TABLES OF EQUIVALENCE CLASSES OF IRREDUCIBLE POLYNOMIALS -----	42
COMPUTER PROGRAMS	-----	46
BIBLIOGRAPHY	-----	79
INITIAL DISTRIBUTION LIST	-----	80
FORM DD 1473	-----	81

ACKNOWLEDGEMENT

The author wishes to express his sincere appreciation of the guidance and assistance provided by Professor Daniel L. Davis in the research for and preparation of this work. Professor Davis suggested the basic concept of a transformational scheme and contributed invaluable aid throughout the effort.

I. INTRODUCTION

A polynomial is said to be reducible in the ring of polynomials with integer coefficients, called $Z[x]$, if it is expressible as the product of polynomials in $Z[x]$. A polynomial not reducible as defined above is irreducible in $Z[x]$.

The polynomials under consideration in this research were those polynomials in $Z[x]$ and reducibility over the ring of integers, Z . The general form of this type of polynomial will be stated as follows:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (1)$$

Investigation into the irreducibility aspects of polynomials leads to two primary avenues of research. The investigator may attempt to determine whether a given polynomial is irreducible or not. The determination is made as the result of some analysis of the polynomial. The analysis may consist of identifying some pattern among the coefficients or some predetermined relationship among the coefficients, or some other type of analysis.

The problem then becomes one of finding the appropriate specification for the given polynomial or class of polynomials which will resolve the irreducibility of the polynomial or class of polynomials under investigation. These specifications for individual polynomials or classes of polynomials are known as 'irreducibility criteria'. Several criteria are

known and apply to many different polynomials (Dorwart, 1935). However, no general irreducibility criterion has been found which will apply to all polynomials in $Z[x]$.

Alternatively, the investigator may attempt to extract the factors of a given polynomial directly without first deciding the question of irreducibility. If the process results in the extraction of factors then the polynomial is reducible; otherwise, it is irreducible. This approach has resulted in several algorithms which extract certain types of factors (Knuth, 1969). The Kronecker method (Dorwart, 1935) is a general solution to the problem of factor extraction of polynomials in $Z[x]$ but it is awkward to apply and the calculations involved are prohibitive in length so that the method is of little utility for practical work. An illustration of this point will be discussed later.

Thus, both paths of attack into the area of irreducible polynomials in $Z[x]$ have been less than completely successful. A general irreducibility criterion, if one exists, has not been identified and the general solution to factor extraction is prohibitive in length even with the speed of calculation provided by the electronic computer.

II. TRANSFORMATIONAL SCHEME

After reviewing several studies in the area of irreducible polynomials in $\mathbb{Z}[x]$, it was decided that rather than attempt to solve the problem of irreducibility directly for a given polynomial, perhaps it would be more fruitful to attempt to facilitate the solution by somehow transforming the given polynomial into a polynomial for which the irreducibility had previously been decided, or would be easier to decide.

This decision as to a method of attack led to a search for transformations which could be used to, in some sense, simplify the polynomials under investigation. To be of any utility for this work, the transformations would have to be such that the irreducibility qualities of a polynomial would be preserved across the transformation.

The first transformation which suggested itself was to transform an arbitrary polynomial with integer coefficients, say $f(x)$, to a primitive polynomial, i.e., a polynomial, say $g(x)$, such that the greatest common divisor of the coefficients is unity. The irreducibility of $f(x)$ is preserved since $f(x)$ is the product of $g(x)$ and some integer k , which is the gcd of the coefficients of $f(x)$. Clearly, if $f(x)$ is irreducible, then so is $g(x)$ and vice versa.

Given a primitive polynomial, it may be further simplified by transforming it to a polynomial whose leading coefficient is unity, known as a monic polynomial. Consider a primitive polynomial with integer coefficients

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (2)$$

where $a_n \neq 1$. Then a transformation [3] called M, such that

$$p(x) \xrightarrow{M} a_n^{n-1} p\left(\frac{x}{a_n}\right) = q(x) \quad (3)$$

and

$$(4)$$

$$q(x) = x^n + a_{n-1} x^{n-1} + a_n a_{n-2} x^{n-2} + \dots + a_n^{n-2} a_1 x + a_n^{n-1} a_0$$

produces a monic polynomial with integer coefficients (MPWIC).

The question arises as to the preservation of irreducibility across this transformation.

Assume that $p(x)$ is irreducible in $Z[x]$ and suppose that $q(x)$ is reducible in $Z[x]$. Note that $q(x)$ is monic, hence primitive, then

$$q(x) = k(x) r(x) \quad (5)$$

$$a_n^{n-1} f\left(\frac{x}{a_n}\right) = k(x) r(x) \quad (6)$$

$$f\left(\frac{x}{a_n}\right) = \frac{1}{a_n^{n-1}} k(a_n x) r(a_n x). \quad (7)$$

Substituting $a_n x$ for x :

$$f(x) = \left(\frac{1}{a_n^{n-1}} \right) k(a_n x) r(a_n x) \quad (8)$$

hence $f(x) = k'(x) r'(x)$ where $k'(x)$ and $r'(x)$ are in $Q[x]$.

But Gauss' theorem states that $f(x)$ is reducible in $Q[x]$ if and only if it is reducible in $Z[x]$. Therefore we have a

contradiction and must conclude that if $p(x)$ is irreducible then $q(x)$ is also irreducible. This result establishes the preservation of irreducibility and gives the result that if $p(x)$ is a primitive irreducible polynomial with integer coefficients then $q(x) = a_n^{n-1} p(x/a_n)$ is a monic irreducible polynomial with integer coefficients (MIPWIC). Since both the reduction to primitivity and the transformation to monicity preserve irreducibility, it follows that any polynomial may be transformed to a MPWIC with irreducibility properties identical to the original polynomial. Thus the only polynomials to be investigated for irreducibility are monic polynomials with integer coefficients.

Attempts were made to discover other transformations which would further simplify the MPWICS's, while preserving monicity, irreducibility, and integral coefficients. One transformation considered was

$$f(x) \longrightarrow f(x+k) \quad , \quad k \text{ in } \mathbb{Z} \quad (9)$$

where if

$$f(x) = x^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0 \quad (10)$$

then

$$f(x+k) = (x+k)^n + a_{n-1}(x+k)^{n-1} + \dots + a_1(x+k) + a_0. \quad (11)$$

Clearly, the transformation preserves monicity and integral coefficients. As to the preservation of irreducibility, suppose that $f(x)$ is irreducible and $f(x+k)$ is reducible. Consider $f(x+k) = g(x) h(x)$. Replacing x by $x-k$ we get the expression

$$f(x) = g(x-k) h(x-k) \quad (12)$$

$$f(x) = g'(x) h'(x) \quad (13)$$

where $g'(x)$ and $h'(x)$ are in $Z[x]$. Therefore $f(x+k) = g(x) h(x)$ implies that $f(x) = g(x-k) h(x-k)$ which is a contradiction of the assumption that $f(x)$ is irreducible. Hence, irreducibility is preserved across the transformation. This transformation will be subsequently referred to as the 'T' transformation. Other notation used to describe the transformation will be

$$f(x) \xrightarrow{T_k} g(x) \quad \text{and} \quad fT_k = g. \quad (14)$$

The next transformation considered will be known as the 'D' transformation in that its effect on a polynomial is a dilation of the axes. The D transformation is specified by

$$f(x) \longrightarrow b^n f\left(\frac{x}{b}\right), \quad n = \deg(f(x)). \quad (15)$$

Using similar notation as for the T transformation, other forms are

$$f(x) \xrightarrow{D_b} g(x) \quad \text{and} \quad fD_b = g.$$

The effect of this transformation is given by the following expression. If

$$f(x) = x^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0 \quad (16)$$

then (17)

$$b^n f\left(\frac{x}{b}\right) = x^n + a_{n-1}bx^{n-1} + \dots + a_1b^{n-1}x + a_0b^n.$$

Clearly, monicity is preserved. If the number b is an integer then the transformation preserves integral coefficients. However, this number may be the reciprocal of an integer if the coefficients of $f(x)$ are such that b^m divides a_{n-m} for each coefficient of $f(x)$. The preservation of irreducibility is proven by methods analogous to the monicity transformation using Gauss' theorem.

A third transformation, called the X transformation, was considered. This transformation consisted of replacing the variable in $f(x)$ by its reciprocal and multiplying the resulting polynomial by x^n . This transformation was shown to preserve integral coefficients after rationalization but preserved monicity only in the case where the last coefficient of $f(x)$ was unity since its effect on a polynomial is to reverse the order of the coefficients. Thus the X transformation was eliminated from further consideration as being too limited in application for the purposes of this work.

Noting that each different value of k or b for the T and D transformations will produce a unique irreducible polynomial it is clear that a single irreducible polynomial generates an infinite number of irreducible polynomials. This leads to the idea of 'equivalence classes' of polynomials with an equivalence class being defined as a set of irreducible polynomials of like degree, each of which could be produced from any other member of the set through a properly chosen sequence of D or T transformations.

Several relationships were developed which relate the effects of sequential series of the two transformations.

$$f(x) \xrightarrow{T_{k_1}} f(x+k_1) \xrightarrow{T_{k_2}} f(x+k_1+k_2) \quad (18)$$

but, if $k' = k_1 + k_2$, then

$$f(x) \xrightarrow{T_{k'}} f(x+k') = f(x+k_1+k_2). \quad (19)$$

Thus, the effect of any pair of successive T transformations can be produced by a single T transformation with a properly chosen value of k. Through repeated applications of Equation (19), it is clear that any finite sequence of T transformations can be replaced by a single T transformation.

Similarly,

$$f(x) \xrightarrow{D_{b_1}} b_1^n f\left(\frac{x}{b_1}\right) \xrightarrow{D_{b_2}} b_1^n b_2^n f\left(\frac{x}{b_1 b_2}\right) \quad (20)$$

but, if $b_3 = b_1 b_2$, then

$$f(x) \longrightarrow b_3^n f\left(\frac{x}{b_3}\right) = b_1^n b_2^n f\left(\frac{x}{b_1 b_2}\right). \quad (21)$$

Therefore, the effect of any two successive D transformations can be achieved by a single D transformation through the proper choice of the transformation factor.

Now, consider successive transformations of different types, i.e., a D transformation followed by a T transformation, or vice versa. Consider the $T_k D_b$ pair in the following expression:

$$f(x) \xrightarrow{T_k} f(x+k) \xrightarrow{D_b} b^n f\left(\frac{x}{b} + k\right). \quad (22)$$

But, after rationalization, the result is

$$f(x) \xrightarrow{D_b} b^n f\left(\frac{x}{b}\right) \xrightarrow{T_k} b^n f\left(\frac{x+k}{b}\right) . \quad (24)$$

In this case, simple algebraic manipulation yields the following expression:

$$f(x) \xrightarrow{D_b T_k} b^n f\left(\frac{x}{b} + \frac{k}{b}\right) , \quad (25)$$

which is the same result as would be affected by the $T_{k/b} D_b$ transformation pair. From the establishment of the two pair relationships

$$T_k D_a = D_a T_{ka} \quad (26)$$

and

$$D_a T_k = T_{\frac{k}{a}} D_a , \quad (27)$$

along with Equations (19) and (21), it follows that any finite sequence of D and T transformations may be collapsed into a DT or TD pair of transformations, by first collapsing the runs of like transformations and then reversing unlike pairs to produce successive like pairs which may again be collapsed into single transformations. The reduction process may be continued until the effect of the sequence is represented by, at most, a TD or DT pair of transformations.

Single transformations, either T or D, are included in the case of pairs since, from the definition of the transformations, it follows that a T_0 or D_1 transformation returns the original polynomial. Thus, any single transformation may be considered to be part of a DT or TD pair, where the other

transformation in the pair is the identity transformation for the unspecified type of transformation. Note also that each type of transformation has its inverse. The inverse of the T_k transformation is T_{-k} and the inverse of the D_a transformation is $D_{1/a}$.

Summarizing, a transformational scheme has been developed, and an algebra of the operators established, which will permit the manipulation of the polynomials under investigation in this research.

III. DEVELOPMENT OF ALGORITHMS FOR TRANSFORMATIONAL OPERATIONS

The transformational scheme described in Section II establishes a coherent framework for manipulation of polynomials. However, in order to actually perform these operations on a digital computer, a series of algorithms must be developed. Further, to be of more than nominal value, the algorithms have to be general, i.e., perform the desired operation on a monic polynomial with integer coefficients of degree n where n is a positive integer greater than or equal to 2.

The first operation to be considered is the single D transformation. By definition

$$f(x) \xrightarrow{D_b} b^n f\left(\frac{x}{b}\right) \quad (28)$$

Let

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (29)$$

then

$$b^n f\left(\frac{x}{b}\right) = b^n \left[a_n \left(\frac{x}{b} \right)^n + a_{n-1} \left(\frac{x}{b} \right)^{n-1} + \dots + a_1 \left(\frac{x}{b} \right) + a_0 \right] \quad (30)$$

$$= a_n x^n + a_{n-1} b x^{n-1} + \dots + a_1 b^{n-1} x + a_0 b^n \quad (31)$$

$$= \sum_{i=0}^n a_{n-i} b^i x^{n-i} \quad (32)$$

Thus, Equation (32) establishes an algorithm for computing the coefficients of $b^n f(x/b)$ given the coefficients of $f(x)$,

the degree n , and the value of b . Programming of the algorithm is straightforward and will not be discussed further.

An algorithm for computing the coefficients of the result of a T transformation was more difficult since, except for the leading coefficient, there is no simple relationship between the coefficients of the original polynomial and the transformed polynomial. The definition of the T transformation gives the expression

$$f(x) \xrightarrow{T_k} f(x+k) . \quad (33)$$

Let

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (34)$$

then

$$f(x+k) = a_n (x+k)^n + a_{n-1} (x+k)^{n-1} + \dots + a_1 (x+k) + a_0 . \quad (35)$$

For an algorithmic solution, consider Taylor's formula, as given by the expression

$$f(x) = \frac{f(a)}{0!} + \frac{f'(a) (x-a)}{1!} + \dots + \frac{f^{(n)}(a) (x-a)^n}{n!} . \quad (36)$$

Substituting $x+k$ for x and letting $k=a$, Taylor's formula becomes

$$f(x+k) = \frac{f(k)}{0!} + \frac{f'(k) x}{1!} + \dots + \frac{f^{(n)}(k) x^n}{n!} \quad (37)$$

which yields the interesting result that the coefficient, b_i , of $f(x+k)$ is given by

$$b_i = \frac{f^{(i)}(k)}{i!} \quad (38)$$

Since

$$f(k) = a_n k^n + a_{n-1} k^{n-1} + a_{n-2} k^{n-2} + \dots + a_1 k + a_0 \quad (39)$$

then

$$f'(k) = a_n n k^{n-1} + a_{n-1} (n-1) k^{n-2} + a_{n-2} (n-2) k^{n-3} + \dots + a_1 \quad (40)$$

⋮

$$f^{(n-m)}(k) = a_n (n)(n-1) \dots (n-1+1) k^m + a_{n-1} (n-1)(n-2) \dots (n-1) k^{m-1} + \dots + a_{n-m+1} \quad (41)$$

⋮

$$f^{(n-1)}(k) = a_n (n)(n-1) \dots (2) k + a_{n-1} (n-1)(n-2) \dots (1) \quad (42)$$

$$f^{(n)}(k) = a_n (n)(n-1) \dots (1). \quad (43)$$

Stating the general term of the sequence in series notation gives the expression

$$f^{(n-m)}(k) = \sum_{i=0}^m a_{n-i} \frac{(n-i)! k^{m-i}}{(m-i)!} \quad (44)$$

Substituting Equation (44) into Equation (39) for $f^{(n-m)}(k)$:

$$f(x+k) = \sum_{j=0}^n \frac{\left(\sum_{i=0}^j a_{n-i} \frac{(n-i)! k^{j-i}}{(j-i)!} \right) x^{n-j}}{(n-j)!} \quad (45)$$

Equation (45) provides an algorithm for computing the coefficients of $f(x+k)$ given the coefficients of $f(x)$, the degree n of $f(x)$, and a value of k . Again, as in the D transformation algorithm, the programming is straightforward except for the computation of the necessary factorials, which can be handled by an iterative or recursive function.

The development of the algorithm for the T and D transformations permits the transformation by computer of any polynomial by any value of k and any non-zero value of a . However, since the primary goal of the research, hence, transformational scheme, is to reduce sets of polynomials into classes rather than generate classes of polynomials, investigation of methods of determining whether a transformation exists between two arbitrary polynomials of like degree is of greater interest.

The cases of two polynomials differing by a single transformation or by a sequence of like transformations will not be discussed separately as the relationship of a single transformation to a pair of transformations was discussed in Section II and an algorithm for a pair will include the cases of single transformations. As has previously been established, any sequence of T and D transformations can readily be collapsed into a DT or TD pair and, further, a TD pair can be replaced by a DT pair of transformations and vice versa. Thus, any algorithm to determine the transformational relationship between two polynomials of like degree need only determine if they are related by, at most, a pair of unlike

transformations. Rather than test for both types of pairs an algorithm which determines only a DT or TD relationship is sufficient since the existence of one pair implies the existence of the other pair between the same two polynomials.

On examination of Equations (26) and (27), which established the relationship of the two pairs for any two polynomials, note that in the DT pair the total translation factor is either k or ka . Hence, if k and a are both integers, the total translational factor of the pair is also an integer. Also, from the definition of the D transformation, note that the factor, a , must be either an integer or the reciprocal of an integer if the coefficients of the transformed polynomial are to be integers. From the statement of the inverse of the D transformation, it is to be noted that if the $D_a T_k$ or $D_a T_{ka}$ transformation pair, whichever exists, from $f(x)$ to $g(x)$ is such that a is not an integer, then the $D_a T_k$ or $D_a T_{ka}$ pair, respectively, from $g(x)$ to $f(x)$ will be such that a is an integer. Then, since k must always be an integer, between any two polynomials which are related through a sequence of transformations, there exists a DT pair such that the T factor and the D factor are both integers. That is, of the four relationships

$$f(x) \xrightarrow{D_a T_k} g(x) \iff f(x) \xrightarrow{D_a T_k} g(x) \quad (46)$$

$$f(x) \xrightarrow{D_a T_{ka}} g(x) \iff f(x) \xrightarrow{T_k D_a} g(x) \quad (47)$$

$$g(x) \xrightarrow{D_a T_{ka}} f(x) \iff g(x) \xrightarrow{T_k D_a} f(x) \quad (48)$$

$$g(x) \xrightarrow{D_a T_k} f(x) \iff g(x) \xrightarrow{D_a T_k} f(x) \quad (49)$$

restated here for clarity, at least one must be such that both transformational factors are integers in the DT pair.

Hence, the algorithm has only to determine whether or not at most a DT pair with integral factors exists between two polynomials to completely decide whether they are related by any sequence of transformations.

Let

$$f(x) = b_n x^n + b_{n-1} x^{n-1} + \dots + b_1 x + b_0 \quad (50)$$

and

$$g(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_1 x + c_0 \quad (51)$$

be two polynomials such that

$$f(x) \xrightarrow{D_a T_k} a^n f \frac{x+k}{a} = g(x) . \quad (52)$$

Substitution in Equation (52) gives the expression

$$a^n b_n \frac{x+k}{a}^n + b_{n-1} \frac{x+k}{a}^{n-1} + \dots + b_1 \frac{x+k}{a} + b_0 = c_n x^n + c_{n-1} x^{n-1} + \dots + c_1 x + c_0 . \quad (53)$$

By repeated differentiation of both sides, we obtain

$$a^n f^{(n-2)} \left(\frac{x+k}{a} \right) = a^n b_n \left(\frac{n!}{2} \right) \left(\frac{1}{a^{n-2}} \right) \left(\frac{x+k}{a} \right)^2 + b_{n-1} (n-1)! .$$

$$\begin{aligned}
\cdot \left(\frac{1}{a^{n-2}} \right) \left(\frac{x+k}{a} \right) + b_{n-2}(n-2)! &= c_n \left(\frac{n!}{2} \right) x^2 + c_{n-1}(n-1)! x \\
+ c_{n-2}(n-2)! &= g^{(n-2)}(x) .
\end{aligned} \tag{54}$$

and

$$\begin{aligned}
a^n b_n (n-1)! \left(\frac{x+k}{a} \right) &= a^n b_n n! \left(\frac{1}{a^{n-1}} \right) \left(\frac{x+k}{a} \right) + b_{n-1}(n-1)! \left(\frac{1}{a^{n-1}} \right) = \\
c_n(n!) x + c_{n-1}(n-1)! &= g^{(n-1)}(x) \tag{55}
\end{aligned}$$

Since Equations (54) and (55) are true for all values of x , they are certainly true for $x = 0$. Simplifying and evaluating at $x = 0$, it follows that

$$\begin{aligned}
a^2 b_n \left(\frac{n!}{2} \right) \left(\frac{k}{a} \right)^2 + a^2 b_{n-1}(n-1)! \left(\frac{k}{a} \right) + b_{n-2}(n-2)! &= \\
c_{n-2}(n-2)! &
\end{aligned} \tag{56}$$

and

$$a b_n n! \left(\frac{k}{a} \right) + a b_{n-1}(n-1)! = c_{n-1}(n-1)! . \tag{57}$$

Further simplification produces a system of two equations in two unknowns, k and a .

$$k^2 \left[b_n n(n-1) \right] + k \left[2b_{n-1} a(n-1) \right] + \left[2b_{n-2} a^2 - 2c_{n-2} \right] = 0 \tag{58}$$

and

$$b_n k n + b_{n-1} a = c_{n-1} . \tag{59}$$

Solving Equation (59) for a in terms of k and substituting for a in Equation (58), an equation in k is produced which

is easily solvable by the use of the quadratic formula:

$$\begin{aligned}
 & k^2 \left[b_n n b_{n-1}^2 (1-n) + 2b_n^2 n^2 b_{n-2} \right] \\
 & + k \left[2c_{n-1} n (b_{n-1}^2 - 2b_{n-2} b_n - 2c_{n-1} b_{n-1}^2) \right] \quad (60) \\
 & + \left[2b_{n-2} c_{n-1}^2 - 2b_{n-1}^2 c_{n-2} \right] = 0 .
 \end{aligned}$$

If a solution of Equation (60) is a non-negative integer, Equation (59) is solved with this value of k . If a is also an integer, then the $D_a T_k$ transformation is performed on $f(x)$ using the algorithms previously developed and the result is compared to $g(x)$. If $f(x) D_a T_k = g(x)$ then there is a relationship between $f(x)$ and $g(x)$ and the factors of the transformation are k and a from Equations (59) and (60). If neither solution of Equation (60) is satisfactory then Equation (60) is modified by substituting b_i for c_i and c_i for b_i and the result is solved in the same manner. This interchange technique allows testing for an integral pair of transformations from $g(x)$ to $f(x)$. If $g(x) D_a T_k = f(x)$ for some values k and a then again there is a relationship and it is specified. If not, there exists no transformational relationship between $f(x)$ and $g(x)$.

The implementation of the algorithm is straightforward except for the special cases of b_{n-1} and/or c_{n-1} being zero since Equation (59) cannot be solved in these cases. The

technique utilized in these special cases is to solve Equation (59) for k instead of a and then substitute this solution into Equation (60) giving the following system of equations:

$$a^2 \left[b_{n-1} (b_{n-1} - 2) (n-1) + 2b_n b_{n-2} n \right] \quad (61)$$

$$+ a \left[2b_{n-1} (c_{n-1} - 1) (n-1) \right] + \left[c_{n-1}^2 (n-1) - 2b_n c_{n-2} n \right] = 0$$

and

$$b_n k + a b_{n-1} = c_{n-1} \quad (62)$$

Equation (62) is always solvable since the degree n is equal to or greater than 2 and $b_n = 1$ due to the monicity constraint. Clearly, if $b_{n-1} = c_{n-1} = 0$ then $k = 0$ but this is expected since, from Equation (38),

$$b_i = \frac{f^{(i)}(k)}{i!} \quad (63)$$

hence,

$$c_{n-1} = b_n \left(\frac{n!}{2} \right) k + b_{n-1} (n-1)! \quad (64)$$

Thus, since $b_n = 1$, the condition $k = 0$ is the only one which, along with the prior condition that $a_{n-1} = 0$, will allow $b_{n-1} = 0$. Therefore, if both a_{n-1} and b_{n-1} are equal to zero, the transformations, if any, between $f(x)$ and $g(x)$ must be solely of the D type since $k = 0$. This possibility is tested by solving for a D transformation factor from b_0

and c_0 . That is, if $g(x) = a^n f(x/a)$ then $c_0 = a^n b_0$. Solving for integral values of a and then calculating the trial polynomial as in the general case allows the testing of this condition.

With the development of the algorithm for testing $f(x)$ and $g(x)$ pairs, the necessary algorithms have been developed for performing either the D or T transformations on a given polynomial or determining the transformational relationship between two polynomials.

IV. SPECIFICATION AND COLLECTION OF DATA BASE

To test the utility and efficacy of the developed transformational scheme and the associated algorithms as a technique for investigation of irreducible polynomials in $\mathbb{Z}[x]$, a representative data base of irreducible polynomials was necessary. To provide a large but finite number of polynomials from which to select the irreducible polynomials, certain conditions were specified and all polynomials meeting these conditions were selected and then screened for irreducibility. The conditions for selection were as follows:

- (a) $f(x)$ is a monic polynomial with integer coefficients
- (b) $f(k)$ is in \mathbb{Z} for k in \mathbb{Z}
- (c) the absolute value of $f(k)$ is less than or equal to 5 for $k = 0, 1, \dots, n$ where n is the degree of $f(x)$.

Conditions (b) and (c) imply a grid of points in the x - y plane, which is represented graphically in Fig 1. The data base is the set of all polynomials satisfying the three conditions specified above. Note that monicity, by definition, requires that the leading coefficient, a_n , be equal to 1, rather than $|a_n| = 1$. But if a polynomial, say $f(x)$, exists such that it satisfies all the conditions for inclusion except that $a_n = -1$, then there exists another polynomial say $g(x)$, such that $f(x) = -g(x)$ and $g(x)$ will be included in the data base. That is, each of the monic polynomials included in the data set will have a reflection across the

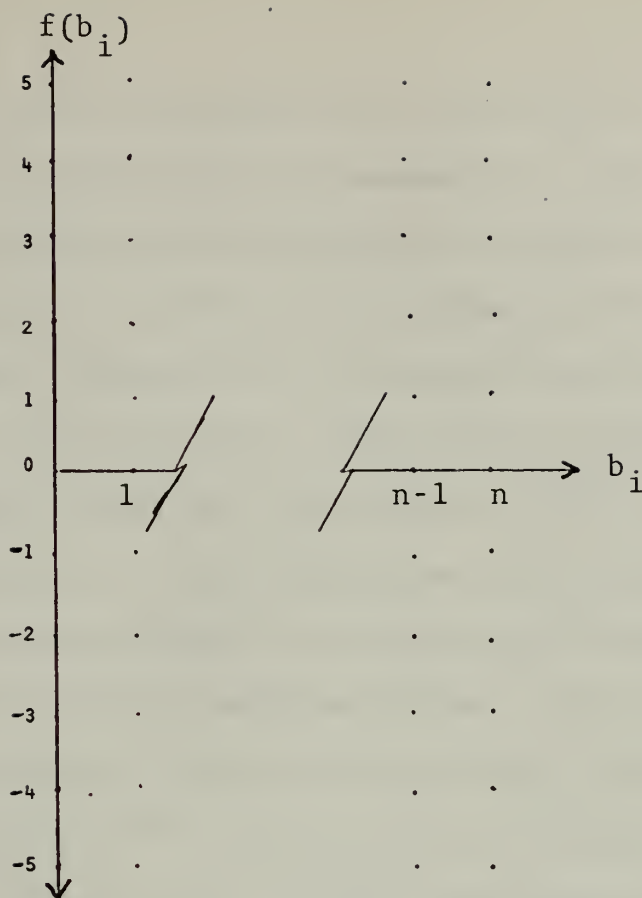


Figure 1.

x-axis which lies on the degree grid but is not included in the data base. But the factorization or irreducibility of the reflected polynomial is identical to that of its reflection with the additional factor of -1 . Hence, the irreducibility or factorization of both a polynomial, $f(x)$, and its reflection, $-f(x)$, are represented by the inclusion of $f(x)$ in the data base.

The actual calculation of all the polynomials for a given degree grid can be accomplished by use of the LaGrange interpolation formula

$$f(x) = \sum_{i=0}^n f(b_i) \frac{(x-b_0) \dots (x-b_{i-1})(x-b_{i+1}) \dots (x-b_n)}{(b_i-b_1) \dots (b_i-b_{i-1})(b_i-b_{i+1}) \dots (b_i-b_n)} \quad (65)$$

where b_i are the abscissae and the $f(b_i)$ are the ordinates of the $n+1$ points for the summation. After the calculation, the polynomial can be screened for monicity and integral coefficients. Clearly, this technique requires that a LaGrangian polynomial be calculated for every possible set of $f(b_i)$ over the grid. Note that the number of sets of functional values has the order of 11^{n+1} where n is the maximum degree of the LaGrangian polynomials. Hence, the number of polynomials to be calculated becomes extremely large for grids of moderate degree. The calculations for each polynomial consist of the actual calculation of the polynomial and then testing the polynomial for monicity and integral coefficients. After programming the LaGrange formula and executing test runs with limited grids, it was determined that approximately 780 microseconds were required for each fifth degree polynomial. Extrapolating to the full grid for the fifth degree, this technique would have required just under 24 hours of central processing unit time on an IBM 360/67 for the processing of the 1,771,561 possible sets of points. Thus an exhaustive search technique, which is analagous to the Kronecker method of factorization, is prohibitive in terms of time required.

Examination of the grid and LaGrangian formula, along with the statement of the problem, offers several areas for reduction of the number of polynomials to be calculated. The previously mentioned case of reflected pairs allows elimination of half the sets of functional values. Thus, if the

the condition that $f(0) \geq 0$ is imposed then one member of each reflected pair is eliminated and the number of polynomials to calculate is halved. Additionally, since the goal is to find irreducible polynomials, any set of points such that at least one $f(b_i) = 0$ may be eliminated since for any k , $f(k) = 0$ implies that k is a root and hence $(x-k)$ is a factor of $f(x)$. Therefore, $f(x)$ is reducible.

Considering the LaGrange formula and actual calculation of the polynomials, note that the requirement that the polynomials be monic imposes constraints on the possible values of $f(b_i)$. The LaGrange formula may be written in the form

$$f(x) = \sum_{i=0}^n f(k) \frac{Y_k(x)}{Y_k(k)} \quad (66)$$

where

$$Y_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^n (x-j). \quad (67)$$

Then

$$f^{(n)}(0) = \sum_{k=0}^n f(k) \frac{Y_k^{(n)}(0)}{Y_k(k)} \quad (68)$$

but, since Y_k is monic by construction

$$Y_k^{(n)}(0) = n! \quad (69)$$

hence

$$f^{(n)}(0) = \sum_{k=0}^n f(k) \frac{n!}{Y_k(k)} . \quad (70)$$

By definition, $Y_k(k) = k(k-1)\dots(1)(-1)\dots(-(n-k))$ thus

$$Y_k(k) = k! (-1)^{n-k} (n-k)! . \quad (71)$$

Substituting Equation (71) into Equation (70), we have the expression

$$f^{(n)}(0) = \sum_{k=0}^n f(k) \frac{n!}{k! (-1)^{n-k} (n-k)!} \quad (72)$$

or

$$f^{(n)}(0) = \sum_{k=0}^n f(k) \binom{n}{k} (-1)^{n+k} \quad (73)$$

But $f^{(n)}(0) = f^{(n)}(x)$ which is just the leading coefficient of $f(x)$. Since the goal is to find monic polynomials then only those sets of points which satisfy

$$1 = \sum_{k=0}^n f(k) \binom{n}{k} (-1)^{n+k} . \quad (74)$$

can possibly be functional values of a monic polynomial.

Thus, only sets of points which satisfy Equation (74) need be considered and the LaGrangian polynomials need be calculated for only these sets of points.

In order to illustrate the savings effected through the use of the condition established by Equation (74), consider

the 1408 minutes of CPU time necessary to calculate and test all the sets for the fifth degree grid. Elimination of the reflected pairs sets and those sets containing a zero still leaves 500,000 sets of points to consider which would require approximately 398 minutes of CPU time. The conditions of $f(b_i) \neq 0$ and Equation (74) were programmed and these 500,000 sets were tested. In this case 551 sets were found in approximately 38 seconds of CPU time. The 551 LaGrange polynomials were then calculated and tested in another 32.5 seconds for a total of 70.5 seconds compared to the 398 minutes required for the exhaustive search technique described above.

The same conditions were applied to the grids for degree 2,3, and 4 and a limit of ± 5 for each case. The results are summarized below:

TABLE I

LIMIT X DEGREE GRID	TOTAL SETS	SETS SATISFYING EQUATION (74) AND $f(b_i) \neq 0$	NUMBER OF MPWIC'S FROM LA- GRANGE FORMULA
5 X 2	500	44	44
5 X 3	5000	253	132
5 X 4	50000	1067	107
5 X 5	500000	551	0

The monic polynomials calculated for each grid (see Appendix A) are a superset of the irreducible MPWIC's for

each grid since they are the set of all the monic polynomials with integer coefficients except those which contain a linear factor of $(x-b_i)$ since those polynomials have already been shown to be reducible. However, polynomials with linear factors of the form $(x+k)$ have not been eliminated. Thus each set of polynomials was further screened by a synthetic division program to eliminate those polynomials which contained this type of linear factor. Since the absolute value of the constant term of any linear factor of a polynomial must be less than or equal to the absolute value of the constant term of the polynomial if the polynomial is reducible in $Z[x]$, the synthetic division program tested those values for each polynomial under test. This test, combined with the previous conditions imposed upon the sets of polynomials, constituted a sufficient condition for irreducibility for the MPWIC's of degree 2 and 3 since if MPWIC's of these degrees are reducible in $Z[x]$, they must contain a linear factor. However, those polynomials of degrees ≥ 4 may not contain a linear factor and still be reducible in $Z[x]$ as the product of two irreducible polynomials of degree ≥ 2 .

Suppose that

$$f(x) = g(x) h(x) \tag{75}$$

then for any value k , $f(k)$ is the product of $g(k)$ and $h(k)$, i.e., $f(k)$ is divisible by $g(k)$ and $h(k)$ for any value of k . If $f(x)$, $g(x)$, and $h(x)$ are all in $Z[x]$, then for an integral

value of k , $f(k)$, $g(k)$, and $h(k)$ are in Z . Also, if $f(k) \neq 0$, then $g(k) \neq 0$ and $h(k) \neq 0$. Now if an integer is the product of two or more integers, none of which is zero, then the absolute value of each of the factors of the product is less than or equal to that of the product. That is, if $f(x) = g(x) h(x)$ then for any value of x

$$|f(x)| \geq |g(x)| \geq 1 \quad (76)$$

and

$$|f(x)| \geq |h(x)| \geq 1 \quad (77)$$

Applying these results to the polynomials under discussion, it follows that if $f(x)$ is a monic polynomial with integer coefficients of degree ≥ 4 , $|f(k)| \leq 5$, and $f(x)$ has no linear factors but is reducible in $Z[x]$, then its factors must be polynomials over a grid of lesser degree with bound equal to 5. In the case of the monic polynomials, $f(x)$, of degree 4 with no linear factors and such that $|f(k)| \leq 5$ for $k = 0, 1, \dots, 4$ then any possible irreducible quadratic factors must be included in the set of irreducible quadratics already determined. Thus, trial division of the quartics with no linear factors by each of the irreducible quadratics will establish the irreducibility of the quartic in question. The actual technique employed was the testing of each coefficient of the quartic successively during the division so that the full division occurred only if the quadratic was actually a factor of the quartic.

The results of the various irreducibility techniques are summarized as follows:

TABLE II.

LIMIT X DEGREE GRID	NUMBER OF MPWIC's	NUMBER WITH NO LINEAR FACTOR	NUMBER OF IRREDUCIBLE NPWIC's
5 X 2	44	43	43
5 X 3	132	126	126
5 X 4	107	107	96

The results as illustrated in Table II complete the establishment of data bases of irreducible polynomials for three separate sets of polynomials upon which the utility of the developed transformational scheme and algorithms can be tested. The irreducible polynomials for each grid are listed in Appendix A.

V. APPLICATION OF TRANSFORMATIONAL SCHEME TO DATA BASE

In order to acquire knowledge of the effect of the transformational scheme on a given data base and hence arrive at some conclusions as to the utility of the scheme, the transformational scheme was applied to the data bases described in Section IV, using a program called "TESTER" which actually implements the testing of polynomial pairs as described in Section III. The execution of TESTER actually separates a set of polynomials into equivalence classes, each of which consists of all the polynomials of the original set which are related to each other through the transformation scheme. The application to the sets of polynomials resulted in the reduction of the 43 quadratic irreducible MPWIC's into 13 equivalence classes, 126 cubic irreducible MPWIC's into 91 equivalence classes, and 96 quartic MPWIC's into 93 equivalence classes. These classes are tabulated in Appendix B.

As a check on these results, an interactive program called "XFORM" was written and executed which actually performs desired sequences of D and T transformations upon a selected polynomial. Independently of the results of "TESTER" the same data bases were manipulated by the use of XFORM and the same results were achieved through trial and error transformation sequences.

Also an interactive program called TESTERA was written which is merely a minor revision of TESTER in that two

polynomials are entered via an interactive terminal and TESTERA determines whether or not a transformational relationship exists and if so, describes the relationship.

VI. RESULTS, CONCLUSIONS, AND HYPOTHESES

This research has shown that any polynomial which, when transformed to a monic polynomial, has integer coefficients and is coincident to the grid for its degree over the range of abscissae of the grid, can be tested for irreducibility by testing it for a transformational relationship to one member of each of the classes of irreducible classes of irreducible MPWIC's for that particular degree grid. If a relationship exists with any of these classes then the polynomial is irreducible. Note that the polynomial is tested for irreducibility and is found to be reducible by default rather than the usual technique of attempting to determine reducibility and finding irreducibility by default.

Since there are infinitely many polynomials which may be tested, the primary result of this research has been the establishment of an irreducibility criterion which can be easily and quickly applied to an infinite set of polynomials. For instance, any quartic monic polynomial, say $f(x)$, such that $|f(x)| \leq 5$ for any 5 consecutive integer arguments can be tested for irreducibility by merely transforming the polynomial by means of the T transformation so that the 5 arguments are coincident to the abscissae of the degree 4, limit 5 grid.

It must be noted that, due to the severely limited grids selected, the utility of the D transformation is

demonstrable only for the data base for the quadratics. That is, the limit constraint of $|f(x)| \leq 5$ precluded the possibility of the existence of $f(x)$ and $g(x)$ such that $g(x) = a^n f(x/a)$ and both $f(x)$ and $g(x)$ being coincident to the grid for degree greater than or equal to three. However, in the quadratic case, this limit restriction does not preclude the possibility and results demonstrating the effectiveness of the D transformation were achieved. The execution of a modified version of TESTER which tested only for T transformational relationships between members of the set of 43 irreducible quadratics resulted in the establishment of 16 equivalence classes under this transformation. As previously noted, the testing for DT pairs of transformations resulted in the establishment of only 13 equivalence classes. Comparison shows that 3 pairs of classes under the T transformation are compressed into 3 classes under the DT transformational pair. Thus, the utilization of the D transformation along with the T transformation extended the capabilities of the techniques.

In view of the results of this work, further research into the technique would seem warranted. This further research, if undertaken, could be to extend the technique in one or more of several possible directions. Perhaps other and more powerful transformations could be identified and included in the transformational scheme. Or modification of the specifications of the various data sets might prove fruitful as this author has identified no conceptual barrier to either extension. Merely relaxing the limit constraint

to some other fixed value, possibly the sum of the degree and some well-chosen constant, should provide interesting results when compared to the results of this work. Alternatively, some variable limit such as $f(x) = x^n$ where n is the degree of $f(x)$ might provide some additional insight into the total problem of a transformational scheme of irreducibility criteria.

APPENDIX A

TABLE 1

COEFFICIENTS OF IRREDUCIBLE QUADRATIC MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID

COEFFICIENTS LISTED AS : $AX^2 + BX + C$

A	B	C	A	B	C	A	B	C
1	-5	1	1	-4	1	1	1	-1
1	-3	1	1	-1	-1	1	-1	1
1	-2	-1	1	0	1	1	-3	-1
1	-4	-1	1	-5	2	1	-4	2
1	0	-2	1	-2	2	1	-1	2
1	-2	-2	1	-3	-2	1	-6	3
1	-5	3	1	1	-3	1	-3	3
1	0	-3	1	-2	3	1	-1	-3
1	-1	3	1	-3	-3	1	-6	4
1	2	-4	1	1	-4	1	-3	4
1	-2	4	1	-1	-4	1	-2	-4
1	-7	5	1	3	-5	1	-5	5
1	2	-5	1	-4	5	1	1	-5
1	-3	5	1	0	-5	1	-2	5
1	-1	-5						

TABLE 2

COEFFICIENTS OF IRREDUCIBLE CUBIC MONIC POLYNOMIALS
 WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID
 COEFFICIENTS LISTED AS : $AX^3 + BX^2 + CX + D$

A	B	C	D	A	B	C	D	A	B	C	D
1	-6	11	-1	1	-7	12	-1	1	-1	-5	1
1	-6	10	-1	1	-7	11	-1	1	-2	-3	1
1	-5	8	-1	1	-6	9	-1	1	-3	-1	1
1	-5	7	-1	1	-6	8	-1	1	-4	1	1
1	-3	0	1	1	-4	5	-1	1	-5	6	-1
1	-5	4	1	1	-4	3	1	1	-3	2	-1
1	-4	3	-1	1	-5	5	1	1	-4	4	1
1	-3	1	-1	1	-4	2	-1	1	-6	7	1
1	-5	6	1	1	-2	-1	-1	1	-3	0	-1
1	-6	8	1	1	-5	7	1	1	-2	-2	-1
1	-7	10	1	1	-6	9	1	1	-1	-4	-1
1	-7	13	-2	1	-8	14	-2	1	-6	11	-2
1	-7	12	-2	1	-6	10	-2	1	-2	-3	-2
1	-5	8	-2	1	-3	-1	2	1	-2	-2	2
1	-6	8	-2	1	-4	2	2	1	-4	4	-2
1	-5	5	-2	1	-5	4	2	1	-3	2	-2
1	-4	3	-2	1	-4	4	2	1	-3	1	-2
1	-5	6	2	1	-2	-1	-2	1	-6	8	2
1	-5	7	2	1	-7	14	-3	1	-8	15	-3
1	-1	-6	3	1	-6	11	-3	1	-7	12	-3
1	-3	-2	3	1	-2	-3	3	1	-5	8	-3
1	-6	9	-3	1	-4	1	3	1	-3	0	3
1	-4	5	-3	1	-5	6	-3	1	-5	4	3
1	-4	3	3	1	-3	2	-3	1	-4	3	-3
1	-6	7	3	1	-5	6	3	1	-2	-1	-3
1	-8	16	-4	1	-7	14	-4	1	-8	15	-4
1	-6	12	-4	1	-1	-6	4	1	-6	11	-4
1	-2	-4	4	1	-5	9	-4	1	-4	0	4
1	-3	-1	4	1	-5	7	-4	1	-4	1	4
1	-4	5	-4	1	-5	6	-4	1	-3	3	-4
1	-4	4	-4	1	-4	3	4	1	-3	2	-4
1	-5	5	4	1	-2	0	-4	1	-3	17	-5
1	-9	18	-5	1	-7	15	-5	1	0	-9	5
1	-7	14	-5	1	-8	15	-5	1	-1	-7	5
1	-6	12	-5	1	-7	13	-5	1	-2	-5	5
1	-1	-6	5	1	-6	11	-5	1	-7	12	-5
1	-3	-2	5	1	-2	-3	5	1	-5	8	-5
1	-6	9	-5	1	-3	-1	5	1	-4	6	-5
1	-5	7	-5	1	-4	1	5	1	-3	0	5
1	-4	5	-5	1	-5	6	-5	1	-5	3	5
1	-4	2	5	1	-3	3	-5	1	-4	4	-5
1	-5	4	5	1	-4	3	5	1	-3	2	-5

TABLE 3

COEFFICIENTS OF IRREDUCIBLE QUARTIC MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS FROM LIMIT X DEGREE GRID

COEFFICIENTS LISTED AS : $AX^4 + BX^3 + CX^2 + DX + E$

A	B	C	D	E	A	B	C	D	E
1	-8	21	-20	1	1	-9	26	-24	1
1	-8	21	-19	1	1	-9	26	-23	1
1	-7	13	-3	-1	1	-8	20	-17	1
1	-9	25	-21	1	1	-7	13	-4	-1
1	-8	20	-16	1	1	-9	25	-20	1
1	-8	18	-9	-1	1	-7	15	-11	1
1	-8	20	-15	1	1	-8	19	-12	-1
1	-7	14	-8	1	1	-8	19	-12	1
1	-9	24	-17	-1	1	-8	19	-13	-1
1	-9	25	-19	-1	1	-8	20	-15	-1
1	-7	13	-5	1	1	-9	25	-20	-1
1	-8	20	-16	-1	1	-7	13	-4	1
1	-10	30	-25	-1	1	-9	25	-21	-1
1	-6	8	1	1	1	-6	7	5	-2
1	-10	31	-29	2	1	-8	21	-20	2
1	-9	26	-24	2	1	-7	13	-3	-2
1	-9	25	-21	2	1	-7	13	-4	-2
1	-8	20	-16	2	1	-8	19	-11	-2
1	-8	19	-13	2	1	-8	19	-12	-2
1	-7	14	-8	2	1	-9	25	-19	-2
1	-7	13	-5	2	1	-9	25	-20	-2
1	-8	20	-16	-2	1	-9	26	-26	3
1	-10	31	-30	3	1	-7	12	0	-3
1	-8	21	-20	3	1	-9	26	-24	3
1	-7	13	-3	-3	1	-8	20	-17	3
1	-9	25	-21	3	1	-8	19	-10	-3
1	-7	14	-6	-3	1	-9	24	-16	-3
1	-8	19	-12	-3	1	-7	14	-8	3
1	-9	25	-19	-3	1	-8	20	-15	-3
1	-7	13	-5	3	1	-8	22	-24	4
1	-6	7	6	-4	1	-10	31	-30	4
1	-7	12	0	-4	1	-9	26	-24	4
1	-7	13	-2	-4	1	-9	25	-22	4
1	-8	18	-8	-4	1	-8	19	-10	-4
1	-8	19	-14	4	1	-9	24	-16	-4
1	-7	14	-8	4	1	-9	25	-18	-4
1	-7	13	-6	4	1	-6	6	9	-5
1	-9	27	-29	5	1	-10	32	-33	5
1	-9	27	-28	5	1	-6	7	6	-5
1	-9	26	-26	5	1	-7	12	1	-5
1	-8	21	-21	5	1	-7	12	0	-5
1	-9	26	-24	5	1	-8	18	-7	-5
1	-7	13	-3	-5	1	-8	20	-17	5
1	-8	18	-8	-5	1	-7	15	-12	5
1	-8	20	-16	5	1	-7	14	-6	-5
1	-8	19	-14	5	1	-8	19	-11	-5
1	-7	14	-9	5	1	-9	24	-16	-5
1	-8	19	-12	-5	1	-7	14	-8	5

APPENDIX B

TABLE 1

EQUIVALENCE CLASSES OF QUADRATIC IRREDUCIBLE MONIC
POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C

1	-5	1
1	-3	-3
1	1	-5
1	-1	-5

1	-4	1
1	-2	-2
1	0	-3

1	1	-1
1	-3	1
1	-1	-1
1	-4	-1
1	-6	4
1	2	-4
1	-2	-4
1	-5	5
1	0	-5

1	-1	1
1	-3	3
1	-2	4

1	-2	-1
1	-4	2
1	0	-2

1	0	1
1	-2	2
1	-4	5
1	-2	5

1	-3	-1
1	-5	3
1	1	-3
1	-1	-3

1	-5	2
1	-3	-2
1	1	-4
1	-1	-4

1	-1	2
1	-3	4

1	-6	3
1	2	-5

1	-1	3
1	-3	5

1	-7	5
1	3	-5

1	-2	3

TABLE 2

EQUIVALENCE CLASSES OF CUBIC IRREDUCIBLE MONIC
POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C	D

1	-7	12	-1
1	-4	1	5

1	-1	-5	1
1	-7	11	-1
1	-4	0	4

1	-2	-3	1
1	-5	4	1

1	-6	9	-1
1	-3	0	3

1	-3	-1	1
1	-6	8	-2

1	-6	8	-1
1	-3	-1	2

1	-4	1	1
1	-1	-4	-1
1	-7	12	-5

1	-3	0	1
1	-6	9	-3

1	-5	5	1
1	-2	-2	2

1	-6	7	1
1	-3	-2	3

1	-2	-1	-1
1	-5	6	-3

1	-3	0	-1
1	-6	9	-5

1	-6	8	1
1	-3	-1	4

1	-2	-2	-1
1	-5	5	-2

1	-6	9	1
1	-3	0	5

1	-7	13	-2
1	-4	2	5

1	-7	12	-2
1	-4	1	4

1	-2	-3	2
1	-5	4	2
*****	*****	*****	*****
1	-4	2	2
1	-7	13	-5
*****	*****	*****	*****
1	-2	-1	-2
1	-5	6	-4
*****	*****	*****	*****
1	-6	8	2
1	-3	-1	5
*****	*****	*****	*****
1	-7	14	-3
1	-4	3	5
*****	*****	*****	*****
1	-7	12	-3
1	-4	1	3
*****	*****	*****	*****
1	-2	-3	3
1	-5	4	3
*****	*****	*****	*****
1	-4	3	3
1	-7	14	-5
*****	*****	*****	*****
1	-6	7	3
1	-3	-2	5
*****	*****	*****	*****
1	-2	-1	-3
1	-5	6	-5
*****	*****	*****	*****
1	-8	16	-4
1	-2	-4	4
1	-5	3	5
*****	*****	*****	*****
1	-7	14	-4
1	-4	3	4
*****	*****	*****	*****
1	-5	9	-4
1	-4	6	-5
*****	*****	*****	*****
1	-8	17	-5
1	-2	-3	5
1	-5	4	5
*****	*****	*****	*****

ALL OTHER CUBIC MIPWIC'S FROM TABLE 2 IN APPENDIX A ARE THE SINGLE MEMBERS OF THEIR RESPECTIVE EQUIVALENCE CLASSES

TABLE 3

EQUIVALENCE CLASSES OF QUARTIC IRREDUCIBLE MONIC
POLYNOMIALS WITH INTEGER COEFFICIENTS

CLASSES ARE SEPARATED BY ROWS OF ASTERISKS

A	B	C	D	E

1	-10	30	-25	-1
1	-6	6	9	-5

1	-10	31	-30	3
1	-6	7	6	-5

1	-6	7	6	-4
1	-10	31	-30	4

ALL OTHER QUARTIC MIPWIC'S FROM TABLE 3 IN APPENDIX A ARE
THE SINGLE MEMBERS OF THEIR RESPECTIVE EQUIVALENCE CLASSES

NOTES AND INSTRUCTIONS ON UTILIZATION OF PROGRAM "XFORM"

1. "XFORM" is a program which performs selected "T" and/or "D" transformations on a desired polynomial.
2. The degree of the polynomial is specified by the entry of a positive integer.
3. The coefficients of the desired polynomial are entered as integers separated by commas, representing the coefficients of the polynomial in descending powers of the variable. Zero coefficients must be included.
4. The type of transformation to be performed is specified by entry of "T" or "D."
5. The translation factor is entered as an integer.
6. The dilation transformation is possible with an integer factor or a factor which is the reciprocal of an integer for certain polynomials. Thus, when the "D" transformation is selected, integer or reciprocal factors must be specified by a response of "I" or "R." Then the actual factor is specified by the entry of an integer value. If the factor is to be an integer, the value is the value entered. If the factor is to be a reciprocal the value is the reciprocal of the integer value entered.
7. The resulting polynomial is displayed as a series of integers separated by commas. These integers are the coefficients of the descending powers of the variable.
8. The "NEXT OPTION?" data entry point allows the operator to select the result of the last transformation, the base

of the last transformation, the original polynomial for the present series of transformations, or the entry of a new polynomial as the next polynomial to be transformed by means of the appropriate response. The responses are "R," "P," "0," and "" respectively.

9. Arithmetic data in the program is represented in binary integers of 31 digits. This allows a range of integers from -262,144 to 262,143 for any single value. This restriction is a limit of the IBM System/360 PL/1 implementation.

10. The output coefficients are displayed to the operator as integers separated by commas. Internally, this is a single character string constructed through various string-handling operations. The maximum length of this string is 72 characters including the commas, again due to the System/360 PL/1 implementations.

11. The maximum number of coefficients for display as output is 64 due to problem data attributes in the program. However, this restriction is insignificant in that the number of coefficients is limited to 36 from Note #9 above.


```

load xform (xeq)
EXECUTION BEGINS...
DEGREE?
_5
COEFFICIENTS SEPARATED BY COMMAS
_1,-1,1,-1,-1,1
TRANSFORMATION TYPE?
_t
K =
_2
1,9,33,61,55,19
NEXT OPTION?
_result
TRANSFORMATION TYPE?
_t
K =
_-2
1,-1,1,-1,-1,1
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_integer
A =
_2
1,-2,4,-8,-16,32
NEXT OPTION?
_r
TRANSFORMATION TYPE?
_d
A= "INTEGER" OR "RECIPROCAL"?
_r
A =
_2
1,-1,1,-1,-1,1
NEXT OPTION?
_
DEGREE?
_4
COEFFICIENTS SEPARATED BY COMMAS
_1,1,1,1,1
TRANSFORMATION TYPE?
_t
K =
_2
1,9,31,49,31
NEXT OPTION?
_(continued on next page)¢

```



```

r
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{2}$ 
1, 18, 124, 392, 496
NEXT OPTION?
 $\bar{r}$ 
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{r}$ 
A =
 $\bar{2}$ 
1, 9, 31, 49, 31
NEXT OPTION?
 $\bar{r}$ 
TRANSFORMATION TYPE?
 $\bar{t}$ 
 $\bar{K}$  =
 $\bar{-2}$ 
1, 1, 1, 1, 1
NEXT OPTION?

 $\bar{D}$ 
DEGREE?
 $\bar{3}$ 
COEFFICIENTS SEPARATED BY COMMAS
 $\bar{1}, 0, 0, 1$ 
*TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{3}$ 
1, 0, 0, 27
NEXT OPTION?
 $\bar{o}$ 
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{-4}$ 
1, 0, 0, -64
NEXT OPTION?

 $\bar{D}$ 
DEGREE?

 $\bar{E}$ 
END OF JOB
R; T=2.11/6.13 13.21.40

```



```

r
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{2}$ 
1, 18, 124, 392, 496
NEXT OPTION?
 $\bar{r}$ 
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{r}$ 
A =
 $\bar{2}$ 
1, 9, 31, 49, 31
NEXT OPTION?
 $\bar{r}$ 
TRANSFORMATION TYPE?
 $\bar{t}$ 
K =
 $\bar{-2}$ 
1, 1, 1, 1, 1
NEXT OPTION?

 $\bar{D}$  DEGREE?
 $\bar{3}$ 
COEFFICIENTS SEPARATED BY COMMAS
 $\bar{1, 0, 0, 1}$ 
*TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{3}$ 
1, 0, 0, 27
NEXT OPTION?
 $\bar{o}$ 
TRANSFORMATION TYPE?
 $\bar{d}$ 
A= "INTEGER" OR "RECIPROCAL"?
 $\bar{i}$ 
A =
 $\bar{-4}$ 
1, 0, 0, -64
NEXT OPTION?

 $\bar{D}$  DEGREE?

 $\bar{E}$  END OF JOB
R; T=2.11/6.13 13.21.40

```



```

XFORM:PROC OPTIONS (MAIN);
/* THIS IS A PROGRAM WHICH APPLIES VARIOUS TRANSFORMATIONS
   TO A POLYNOMIAL AND ALLOWS EXAMINATION OF THE EFFECT OF THESE
   TRANSFORMATIONS. */
/* DECLARE NECESSARY VARIABLES */
DCL BUFFER CHAR(80),
    RESPONSE CHAR(512) VARYING,
    TEMP CHAR(72) VARYING,
    (AMOUNT,DEGREE,TERM,F,I,J,K,L,M,N) FIXED BIN (31),
    NUMBER CHAR(16) VARYING,
    CHOICE CHAR(12) VARYING,
    DIGITS CHAR(10) INITIAL('0123456789'),
    DIGITS1 CHAR(11) INITIAL('0123456789-'),
    STRING1 CHAR(12) INITIAL('0123456789-'),
    ERROR CHAR(24) INITIAL ('INVALID ENTRY----REENTER');

/* REQUEST DESIRED DEGREE AND PROCESS RESPONSE */
TOP:DISPLAY('DEGREE?') REPLY (BUFFER);
RESPONSE='';
DO L = 1 TO 80;
    IF SUBSTR(BUFFER,L,1)~=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);
END;
IF RESPONSE = '' THEN GO TO FINISH;
DO M = 1 TO LENGTH(RESPONSE);
    IF INDEX(DIGITS,SUBSTR(RESPONSE,M,1))=0 THEN DO;
        DISPLAY (ERROR);
        GO TO TOP;
    END;
END;
DEGREE = RESPONSE;
BEGIN;
/* SET UP NECESSARY ARRAYS */
DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));
DCL (TRANSP(0:DEGREE),COEFFS(0:DEGREE),POLY(0:DEGREE))
    FIXED BIN (31);
/* REQUEST COEFFICIENTS AND PLACE RESPONSES INTO PROPER ARRAYS. */
DATA:DISPLAY('COEFFICIENTS SEPARATED BY COMMAS')
    REPLY(BUFFER);

```



```

RESPONSE = '';
DO L = 1 TO 80;
IF SUBSTR(BUFFER,L,1) = ' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);
END;
DO L = 1 TO LENGTH(RESPONSE);
IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;
DISPLAY(SUBSTR(RESPONSE,L,1));
DISPLAY(ERROR);
GO TO DATA;
END;
END;
DO L = 0 TO DEGREE;
IF RESPONSE = '' THEN DO; DISPLAY(ERROR); GO TO DATA; END;
K = INDEX(RESPONSE,'');
IF K=1 THEN DO; DISPLAY(RESPONSE); GO TO DATA; END;
IF K = 0 THEN K = LENGTH(RESPONSE) + 1;
COEFFS(L) = SUBSTR(RESPONSE,1,K-1);
IF INDEX(RESPONSE,'')=0 THEN RESPONSE='';
IF LENGTH(RESPONSE)>K THEN RESPONSE=SUBSTR(RESPONSE,K+1);
ELSE RESPONSE='';
END;
POLY=COEFFS;

/* DETERMINE TYPE OF TRANSFORMATION TO BE PERFORMED */
TRANS:DISPLAY('TRANSFORMATION TYPE?') REPLY (RESPONSE);
IF RESPONSE = '' THEN GO TO TOP;
RESPONSE = SUBSTR(RESPONSE,1,1);
IF RESPONSE = 'T' & RESPONSE = 'D' & RESPONSE = 'M' THEN DO;
DISPLAY(RESPONSE); DISPLAY(ERROR); GO TO TRANS; END;
IF RESPONSE = 'T' THEN CALL TRANS1;
IF RESPONSE = 'D' THEN CALL DILATE;
IF RESPONSE = 'M' THEN CALL MONIC;

TRANS1:PROCEDURE;

/* THE TRANSL PROCEDURE PERFORMS THE TRANSLATION TRANSFORMATION */
/* DETERMINE THE AMOUNT OF TRANSLATION */
TEMP = '';
FACTOR:DISPLAY('K =') REPLY(BUFFER);
RESPONSE='';
DO K = 1 TO 80;
IF SUBSTR(BUFFER,K,1) = ' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,K,1);
END;
DO M = 1 TO LENGTH(RESPONSE);
IF INDEX(DIGITS1,SUBSTR(RESPONSE,M,1)) = 0 THEN DO;

```

```

XF000490
XF000500
XF000510
XF000520
XF000530
XF000540
XF000550
XF000560
XF000570
XF000580
XF000590
XF000600
XF000610
XF000620
XF000630
XF000640
XF000650
XF000660
XF000670
XF000680
XF000690
XF000700
XF000710
XF000720
XF000730
XF000740
XF000750
XF000760
XF000770
XF000780
XF000790
XF000800
XF000810
XF000820
XF000830
XF000840
XF000850
XF000860
XF000870
XF000880
XF000890
XF000900
XF000910
XF000920
XF000930
XF000940
XF000950
XF000960

```



```

      DISPLAY (RESPONSE);
      DISPLAY(ERROR);
      GO TO FACTOR;

      END;
      IF RESPONSE = '' THEN RETURN;
      AMOUNT=RESPONSE;

      /* PERFORM TRANSLATION AND DISPLAY RESULT */

      RESPONSE='';
      DO J = 0 TO DEGREE;
        TERM = 0;
        DO I = 0 TO J;
          IF AMOUNT=0; THEN M = ABS(AMOUNT)**(J-I);
          ELSE DO; TRANSP(J)=COEFFS(J);
            GO TO TAB; END;
          IF AMOUNT < 0 THEN DO; IF MOD(J-I,2) = 1 THEN M = -M;END;
          TERM=TERM+(COEFFS(I)*FACT(DEGREE-I)*M)/FACT(J-I);
        END;
        TRANSP(J) = TERM/FACT(DEGREE-J);
        TAB:PUT STRING(RESPONSE) EDIT(TRANSP(J))(F(8));
        NUMBER='';
        DO N = 1 TO LENGTH(RESPONSE);
          IF SUBSTR(RESPONSE,N,1)=-1 , THEN NUMBER=NUMBER||SUBSTR(RESPONSE,
            N,1);
        END;
        TEMP=TEMP||NUMBER||',';
      END;
      TEMP=SUBSTR(TEMP,1,LENGTH(TEMP)-1);
      DISPLAY(TEMP);

      END. TRANSL;

      DILATE:PROCEDURE;

      /* THE DILATE PROCEDURE PERFORMS THE DILATION TRANSFORMATION */

      /* DETERMINE DILATION FACTOR */

      TEMP = '';
      HEAD:DISPLAY('A= "INTEGER" OR "RECIPROCAL"?') REPLY(CHOICE);
      IF CHOICE = '' THEN DO; DISPLAY (ERROR);
        GO TO HEAD;
      END;
      CHOICE = SUBSTR(CHOICE,1,1);
      IF CHOICE=-1 & CHOICE =-R THEN DO;

```



```

DISPLAY(CHOICE);
DISPLAY(ERROR);
GO TO HEAD;
END;
DISPLAY('A=') REPLY(BUFFER);
RESPONSE='';
DO L=1 TO 80;
  IF SUBSTR(BUFFER,L,1) ^= ' ' THEN RESPONSE = RESPONSE||SUBSTR(BUFFER
    ,L,1);
END;
DO L=1 TO LENGTH(RESPONSE);
  IF INDEX(DIGITS1,SUBSTR(RESPONSE,L,1))=0 THEN DO;
    DISPLAY(RESPONSE);
    DISPLAY(ERROR);
    GO TO HEAD;
  END;
END;
RESPONSE=' ' THEN RETURN;
AMOUNT=RESPONSE;
/* PERFORM THE DILATION TRANSFORMATION AND DISPLAY THE RESULT */
IF AMOUNT = 0 THEN DO; DISPLAY('DILATION FACTOR OF ZERO IS UNDEFINED');
  GO TO HEAD; END;
DO K = 0 TO DEGREE;
  M = ABS(AMOUNT)**(K);
  IF AMOUNT < 0 THEN DO; IF MOD(K,2)=1 THEN M = -M; END;
  IF CHOICE = 'I' THEN TRANSP(K)=COEFFS(K)*M;
  ELSE DO; IF MOD(COEFFS(K),M)=0 THEN TRANSP(K)=COEFFS(K)/M;
    ELSE DO; DISPLAY('NON-INTEGGER COEFFICIENTS');
      GO TO ASK;
    END;
  END;
  PUT STRING (RESPONSE) EDIT (TRANSP(K))(F(8));
  NUMBER='';
  DO N=1 TO LENGTH(RESPONSE);
    IF SUBSTR(RESPONSE,N,1) ^= ' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,
      N,1);
  END;
  TEMP=TEMP||NUMBER||',';
END;
TEMP=SUBSTR(TEMP,1,LENGTH(TEMP)-1);
DISPLAY(TEMP);
END DILATE;
MONIC: PROCEDURE;

```

XF001450
 XF001460
 XF001470
 XF001480
 XF001490
 XF001500
 XF001510
 XF001520
 XF001530
 XF001540
 XF001550
 XF001560
 XF001570
 XF001580
 XF001590
 XF001600
 XF001610
 XF001620
 XF001630
 XF001640
 XF001650
 XF001660
 XF001670
 XF001680
 XF001690
 XF001700
 XF001710
 XF001720
 XF001730
 XF001740
 XF001750
 XF001760
 XF001770
 XF001780
 XF001790
 XF001800
 XF001810
 XF001820
 XF001830
 XF001840
 XF001850
 XF001860
 XF001870
 XF001880
 XF001890
 XF001900
 XF001910
 XF001920


```

/* MONIC TRANSFORMS A GIVEN POLYNOMIAL INTO A MONIC POLYNOMIAL
   VIA THE M TRANSFORMATION. */
TEMP='';
AMOUNT=COEFS(0);
IF AMOUNT = 0 THEN DO;
  DISPLAY('TRANSFORMATION IMPOSSIBLE-LEADING COEFFICIENT ZERO');
  RETURN;
END;
TRANSP(0)=1;
TRANSP(1)=COEFS(1);
DO K = 2 TO DEGREE;
  M=ABS(AMOUNT)**(K-1);
  IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M=-M;
  TRANSP(K)=COEFS(K)*M;
  PUT STRING(RESPONSE) EDIT(TRANSP(K))(F(15));
  NUMBER = '';
  DO N = 1 TO LENGTH(RESPONSE);
    IF SUBSTR(RESPONSE,N,1)='- ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,
      N,1);
  END;
  TEMP=TEMP||NUMBER||',';
END;
NUMBER = '';
DO N = 1 TO LENGTH(CHAR(TRANSP(0)));
  IF SUBSTR(CHAR(TRANSP(0)),N,1)='- ' THEN NUMBER=NUMBER||SUBSTR(
    CHAR(TRANSP(0)),N,1);
END;
NUMBER=NUMBER||',';
DO N=1 TO LENGTH(CHAR(TRANSP(1)));
  IF SUBSTR(CHAR(TRANSP(1)),N,1)='- ' THEN NUMBER = NUMBER||SUBSTR(
    CHAR(TRANSP(1)),N,1);
END;
TEMP = NUMBER||','||SUBSTR(TEMP,1,LENGTH(TEMP)-1);
DISPLAY(TEMP);
END MONIC;

FACT:PROCEDURE (N) FIXED BIN (31) RECURSIVE;
/* THE FACT PROCEDURE COMPUTES N FACTORIAL FOR A GIVEN N */
DCL (F,N) FIXED BIN (31);
IF N <= 0 THEN GO TO SKIP;
F=1;
RETURN (F);
SKIP:N=N-1;
F=FACT(N);
N=N+1;

```

XF001930
 XF001940
 XF001950
 XF001960
 XF001970
 XF001980
 XF001990
 XF002000
 XF002010
 XF002020
 XF002030
 XF002040
 XF002050
 XF002060
 XF002070
 XF002080
 XF002090
 XF002100
 XF002110
 XF002120
 XF002130
 XF002140
 XF002150
 XF002160
 XF002170
 XF002180
 XF002190
 XF002200
 XF002210
 XF002220
 XF002230
 XF002240
 XF002250
 XF002260
 XF002270
 XF002280
 XF002290
 XF002300
 XF002310
 XF002320
 XF002330
 XF002340
 XF002350
 XF002360
 XF002370
 XF002380
 XF002390
 XF002400


```

F=F*N;
RETURN (F);
END FACT;

/* DETERMINE NEXT PROGRAM OPTION */
ASK:DISPLAY('NEXT OPTION?') REPLY(RESPONSE);
IF RESPONSE=' ' THEN GO TO TOP;
IF RESPONSE=SUBSTR(RESPONSE,1,1);
IF RESPONSE = 'R' THEN DO; COEFFS=TRANSP;
GO TO TRANS;
END;
IF RESPONSE='P' THEN GO TO TRANS;
IF RESPONSE = 'O' THEN DO; COEFFS=POLY; GO TO TRANS; END;
IF RESPONSE='R' & RESPONSE='P' & RESPONSE='O' THEN DO;
DISPLAY(ERROR); GO TO ASK; END;
END; /*END OF BEGIN BLOCK */
FINISH:DISPLAY('END OF JOB');
END XFORM;

```

```

XF002410
XF002420
XF002430
XF002440
XF002450
XF002460
XF002470
XF002480
XF002490
XF002500
XF002510
XF002520
XF002530
XF002540
XF002550
XF002560
XF002570
XF002580
XF002590
XF002600
XF002610

```


NOTES AND INSTRUCTIONS ON USE OF PROGRAM "TESTERA"

1. "TESTERA" is a program which determines the transformational relationship between two primitive polynomials with integer coefficients.
2. The "LIST" data entry specifies whether a single polynomial is to be tested against the first polynomial entered or a succession of polynomials to be tested against the same polynomial. Response is 'Y' (yes) or 'N' (no). If "LIST" is specified, the program returns to the entry point for the second polynomial after completion of testing of each polynomial.
3. The "DEGREE?" is entered as a positive integer.
4. Coefficients of polynomials are entered as integers separated by commas representing coefficients of descending powers of variable. Zero coefficients must be included.
5. Results of tests are displayed as integer values of K and A for transformations between the polynomials or, if no transformational relationship exists, "NO TRANSFORMATION" is displayed.
6. If either or both polynomials are non-monic, they are transformed into monic polynomials prior to testing for relationship between them.
7. Size and implementation restrictions for "TESTERA" are the same as for "XFORM."


```

load testera (xeq)
EXECUTION BEGINS...
LIST?
_no
DEGREE?
_4
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,0,0,54,567
NEXT COEFFICIENTS
_1,0,0,2,7
K=0...A=3***
DEGREE?

LIST?
_yes
DEGREE?
_2
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,1,1
NEXT COEFFICIENTS
_1,5,7
K=2...A=1
NEXT COEFFICIENTS
_1,3,9
K=0...A=3
NEXT COEFFICIENTS
_1,-1,1
K=-1...A=1
NEXT COEFFICIENTS
_1,1,2
NO TRANSFORMATIONS
NEXT COEFFICIENTS

LIST?
_no
DEGREE?
_12
FIRST COEFFICIENTS SEPARATED BY COMMAS
_1,1,1,1,1,1,1,1,1,1,1,1
NEXT COEFFICIENTS
_1,2,4,8,16,32,64,128,256,512,1024,2048,4096
K=0...A=2
DEGREE?

LIST?

END OF JOB
R; T=1.73/4.07 13.40.06

```



```

TESTERA:PROCEDURE OPTIONS(MAIN);
    /* TESTERA IS A PROCEDURE WHICH ACCEPTS TWO INTEGRAL POLYNOMIALS
    OF EQUAL DEGREE AND DETERMINES THE T AND D TRANSFORMATION
    RELATIONS BETWEEN THEM. */

    DCL RESPONSE CHAR(512) VARYING INITIAL ('');
    DCL BUFFER CHAR(80);
    DCL DEGREE FIXED BIN (31); VARYING;
    DCL (TEMP,TEMP1) CHAR(72) VARYING;
    DCL NUMBER CHAR(15) VARYING;
    DCL ANSWER CHAR(15) VARYING;
    DCL (KK1,K1) FIXED BIN(31);
    DCL DIGITS CHAR(10) INITIAL ('0123456789');
    DCL DIGITS1 CHAR(11) INITIAL ('0123456789-');
    DCL STRING1 CHAR(12) INITIAL ('0123456789-');
    DCL (L,J,K,FLAG) FIXED BIN (31);
    DCL (A1,A2) FIXED BIN (31);
    DCL TERM FIXED BIN (31);
    DCL DISC FIXED BIN (31.5);
    DCL CHECK FIXED BIN (15);
    DCL ERROR CHAR(25) INITIAL ('INVALID RESPONSE--REENTER');
    DCL CHOICE CHAR(4) VARYING INITIAL ('');

    /* DETERMINE WHETHER INPUT IS INDIVIDUAL PAIRS OR A LIST
    TO BE TESTED AGAINST A SINGLE POLYNOMIAL. */

    FIRST:DISPLAY('LIST?') REPLY(CHOICE);
    IF CHOICE=' ' THEN GO TO FINISH;
    CHOICE=SUBSTR(CHOICE,1,1);
    IF CHOICE='Y' & CHOICE='N' THEN DO; DISPLAY (CHOICE);
    DISPLAY(ERROR);
    GO TO FIRST; END;

    /* DETERMINE DEGREE OF POLYNOMIALS. */

    START:RESPONSE='';
    DISPLAY('DEGREE?')REPLY(BUFFER);
    DO L=1 TO 80;
    IF SUBSTR(BUFFER,L,1)='- ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,
    L,1);
    END;
    IF RESPONSE = ' ' THEN GO TO FIRST;
    DO L=1 TO LENGTH(RESPONSE);
    IF INDEX(DIGITS,SUBSTR(RESPONSE,L,1))=0 THEN DO;
    DISPLAY(SUBSTR(RESPONSE,L,1));
    DISPLAY(ERROR);
    GO TO START;
    END;

```



```

END;
DEGREE = RESPONSE;
BEGIN;

/* DECLARE ARRAYS OF PROPER SIZE FOR DEGREE. */
DCL COEFS(2,0:DEGREE) FIXED BIN (31);
DCL QDAD(3) FIXED BIN (31);

/* REQUEST COEFFICIENTS OF FIRST POLYNOMIAL AND PROCESS INPUT.*/
TOP:DISPLAY('FIRST COEFFICIENTS SEPARATED BY COMMAS')
REPLY (BUFFER);
RESPONSE='';
DO L = 1 TO 80;
  IF SUBSTR(BUFFER,L,1)~=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,
L,1);
END;
DO L = 1 TO LENGTH(RESPONSE);
  IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;
    DISPLAY(SUBSTR(RESPONSE,L,1));
    DISPLAY(ERROR); GO TO TOP; END;
END;
DO L = 0 TO DEGREE;
  IF RESPONSE=''; THEN GO TO FIRST;
  K=INDEX(RESPONSE,'');
  IF K=1 THEN DO; DISPLAY(RESPONSE);DISPLAY(ERROR);GO TO TOP; END;
  IF K=0 THEN K=LENGTH(RESPONSE)+1;
  COEFS(1,L)=SUBSTR(RESPONSE,1,K-1);
  IF INDEX(RESPONSE,'')=0 THEN RESPONSE='';
  IF LENGTH(RESPONSE)>K THEN RESPONSE=SUBSTR(RESPONSE,K+1);
  ELSE RESPONSE='';
END;

/* NOW REQUEST SECOND SET OF COEFFICIENTS AND PROCESS INPUT. */
SECOND:DISPLAY('NEXT COEFFICIENTS') REPLY(BUFFER);
RESPONSE='';
DO L = 1 TO 80;
  IF SUBSTR(BUFFER,L,1)~=' ' THEN RESPONSE=RESPONSE||SUBSTR(BUFFER,L,1);
END;
DO L = 1 TO LENGTH(RESPONSE);
  IF INDEX(STRING1,SUBSTR(RESPONSE,L,1))=0 THEN DO;
    DISPLAY(SUBSTR(RESPONSE,L,1));DISPLAY(ERROR); GO TO SECOND; END;
END;
DO L = 0 TO DEGREE;
  IF RESPONSE=''; THEN GO TO FIRST;

```



```

K=INDEX(RESPONSE,'');
IF K=1 THEN DO; DISPLAY(RESPONSE); GO TO SECOND;END;
IF K=0 THEN K=LENGTH(RESPONSE)+1;
COEFFS(2,L)=SUBSTR(RESPONSE,1,K-1);
IF INDEX(RESPONSE,'')=0 THEN RESPONSE='';
IF INDEX(RESPONSE,'') > K THEN RESPONSE=SUBSTR(RESPONSE,K+1);
ELSE RESPONSE='';
END;

/* NOW CHECK FOR PROPER DEGREE OF POLYNOMIAL. */
IF COEFFS(1,0)=0 THEN DO; DISPLAY('INCORRECT DEGREE-FIRST POLYNOMIAL');
GO TO START; END;
IF COEFFS(2,0)=0 THEN DO; DISPLAY('INCORRECT DEGREE-SECOND POLYNOMIAL');
GO TO START; END;

/* IF POLYNOMIALS ARE NOT MONIC, THEN TRANSFORM THEM TO
MONIC POLYNOMIALS PRIOR TO TEST FOR TRANSFORMATIONAL
RELATIONSHIP. */
IF COEFFS(1,0) /= 1 THEN DO; FLAG=1; CALL MONIC; END;
IF COEFFS(2,0) /= 1 THEN DO; FLAG=2; CALL MONIC; END;

/* IF BOTH TRACES ARE ZERO THEN TRANSFORMATION, IF ANY,
MUST BE PURELY D TYPE, SO USE SPECIAL TEST FOR THIS
CASE. */
IF COEFFS(1,1)=0 & COEFFS(2,1)=0 THEN DO;
IF MOD(COEFFS(2,DEGREE),COEFFS(1,DEGREE))=0 THEN DO;
L=L+1;
TAB1:L=L+1;
TERM=1;
DO J=1 TO DEGREE;
TERM=TERM*L;
END;
IF TERM < KK1 THEN GO TO TAB1;
IF TERM = KK1 THEN DO;
A1=L;
K1=0;
FLAG=2;
CHECK=1;
GO TO GOTONE;
END;
END;
IF MOD(COEFFS(1,DEGREE),COEFFS(2,DEGREE))=0 THEN DO;
KK1=COEFFS(1,DEGREE)/COEFFS(2,DEGREE);
L=L+1;

```

TES00970
TES00980
TES00990
TES01000
TES01010
TES01020
TES01030
TES01040
TES01050
TES01060
TES01070
TES01080
TES01090
TES01100
TES01110
TES01120
TES01130
TES01140
TES01150
TES01160
TES01170
TES01180
TES01190
TES01200
TES01210
TES01220
TES01230
TES01240
TES01250
TES01260
TES01270
TES01280
TES01290
TES01300
TES01310
TES01320
TES01330
TES01340
TES01350
TES01360
TES01370
TES01380
TES01390
TES01400
TES01410
TES01420
TES01430
TES01440


```

TAB2:L=L+1;
TERM=1; 1 TO DEGREE;
DO J = 1 TO TERM*L;
  TERM = TERM*L;
END;
IF TERM < KK1 THEN GO TO TAB2;
IF TERM = KK1 THEN DO;
  A1 = L;
  K1 = 0;
  FLAG = 4;
  CHECK = 0;
  GO TO GOTONE;
END;
END;
DISPLAY('NO TRANSFORMATIONS');
IF CHOICE = 'Y' THEN GO TO SECOND;
ELSE GO TO FIRST;
END;
CHECK = 1;

/* IF COEFFS(1,1)=0 THEN MUST SOLVE QUADRATIC IN A VICE K. */
IF COEFFS(1,1)=0 THEN DO;
  IF MOD(COEFFS(2,1)/(COEFFS(1,0)*DEGREE))=0 THEN GO TO OTHER;
  K1 = COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
  DISC=((2*COEFFS(1,0)*DEGREE*(K1**2)-(COEFFS(2,1)**2))/
        -(COEFFS(2,1)**2*(DEGREE-1)))/
        (2*COEFFS(1,0)*DEGREE);
  IF DISC < 0 THEN GO TO OTHER;
  DISC = SQRT(DISC);
  IF MOD(DISC,1)~=0 THEN GO TO OTHER;
  A1 = DISC;
  FLAG = 1;
  GO TO GOTONE;
END;

/* OTHERWISE, SOLVE STANDARD QUADRATIC IN K. FIRST, CLACULATE
COEFFICIENTS OF QUADRATIC. */
QUAD(1)=(COEFFS(1,0)*DEGREE*(COEFFS(1,1)**2)*(1-DEGREE))+
          (2*(COEFFS(1,0)**2)*(DEGREE**2)*COEFFS(1,2));
QUAD(2)=(2*COEFFS(2,1)*DEGREE*(COEFFS(1,1)**2)-(2*COEFFS(1,2)
          *COEFFS(1,0)))-(2*COEFFS(2,1)*(COEFFS(1,1)**2));
QUAD(3)=(2*COEFFS(1,2)*(COEFFS(2,1)**2)-(2*(COEFFS(1,1)**2)*
          COEFFS(2,2));

/* CHECK FOR SPECIAL CASES OF QUADRATIC. */

```

TES01450
 TES01460
 TES01470
 TES01480
 TES01490
 TES01500
 TES01510
 TES01520
 TES01530
 TES01540
 TES01550
 TES01560
 TES01570
 TES01580
 TES01590
 TES01600
 TES01610
 TES01620
 TES01630
 TES01640
 TES01650
 TES01660
 TES01670
 TES01680
 TES01690
 TES01700
 TES01710
 TES01720
 TES01730
 TES01740
 TES01750
 TES01760
 TES01770
 TES01780
 TES01790
 TES01800
 TES01810
 TES01820
 TES01830
 TES01840
 TES01850
 TES01860
 TES01870
 TES01880
 TES01890
 TES01900
 TES01910
 TES01920


```

IF QUAD(1)=0 THEN DO;
  IF QUAD(2) = 0 THEN GO TO OTHER;
  IF MOD(QUAD(3),QUAD(2))=0 THEN GO TO OTHER;
  K1=QUAD(3)/QUAD(2);
  GO TO L1;
END;
DISC=(QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
IF DISC < 0 THEN GO TO OTHER;
IF MOD(SQRT(DISC),1) = 0 THEN GO TO OTHER;
TERM = SQRT(DISC);
KK1 = -QUAD(2)-TERM;
IF MOD(KK1,2*QUAD(1))=0 THEN GO TO NEXT;
K1=KK1/(2*QUAD(1));
L1: IF MOD(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE),COEFFS(1,1))=0 THEN
  GO TO NEXT;
  GO TO NEXT;
A1=(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE))/COEFFS(1,1);
FLAG=1;
GO TO GOTONE;

/* TRY OTHER SOLUTION OF QUADRATIC. */
NEXT: IF COEFFS(1,1)=0 THEN DO; A1=DISC;
  FLAG=2;
  GO TO GOTONE; END;

IF QUAD(1)=0 THEN GO TO OTHER;
KK1=-QUAD(2)+TERM;
IF MOD(KK1,2*QUAD(1))=0 THEN GO TO OTHER;
K1 = KK1 / (2*QUAD(1));
IF MOD(COEFFS(2,1),((DEGREE*K1)+(COEFFS(1,1)))=0 THEN GO TO OTHER;
A1 = COEFFS(2,1) / ((DEGREE*K1)+(COEFFS(1,1)));
FLAG = 2;
GO TO GOTONE;

/* FLIP POLYNOMIALS TO CHECK FOR TRANSFORMATIONS IN OTHER
  DIRECTION. */
OTHER: CHECK=0;

/* IF COEFFS(2,1)=0 THEN SOLVE QUADRATIC IN A VICE K. */
IF COEFFS(2,1)=0 THEN DO;
  IF MOD(COEFFS(1,1),(COEFFS(2,0)*DEGREE))=0 THEN GO TO NONE;
  K1 = COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
  DISC=((2*COEFFS(2,0)*DEGREE*(COEFFS(1,2))
    -(COEFFS(1,1)**2)*(DEGREE-1)))/
    (2*COEFFS(2,2)*COEFFS(2,0)*DEGREE);
  IF DISC < 0 THEN GO TO NONE;
  DISC = SQRT(DISC);

```

TES01930
 TES01940
 TES01950
 TES01960
 TES01970
 TES01980
 TES01990
 TES02000
 TES02010
 TES02020
 TES02030
 TES02040
 TES02050
 TES02060
 TES02070
 TES02080
 TES02090
 TES02100
 TES02110
 TES02120
 TES02130
 TES02140
 TES02150
 TES02160
 TES02170
 TES02180
 TES02190
 TES02200
 TES02210
 TES02220
 TES02230
 TES02240
 TES02250
 TES02260
 TES02270
 TES02280
 TES02290
 TES02300
 TES02310
 TES02320
 TES02330
 TES02340
 TES02350
 TES02360
 TES02370
 TES02380
 TES02390
 TES02400


```

IF MOD(DISC,1)~=0 THEN GO TO NONE;
A1 = DISC;
FLAG=3;
GO TO GOTONE;
END;

/* OTHERWISE, SOLVE STANDARD QUADRATIC IN K. */
QUAD(1)=(COEFFS(2,0)*DEGREE*(COEFFS(2,1)**2)*(1-DEGREE))
+ (2*(COEFFS(2,0)**2)*(DEGREE**2)*COEFFS(2,2));
QUAD(2)=(2*COEFFS(1,1)*DEGREE)*(COEFFS(2,1)**2)-(2*COEFFS(2,2)
*COEFFS(2,0))-(2*COEFFS(1,1)*(COEFFS(2,1)**2));
QUAD(3)=(2*COEFFS(2,2)*(COEFFS(1,1)**2)-(2*(COEFFS(2,1)**2)
*COEFFS(1,2));

/* CHECK FOR SPECIAL CONDITIONS IN QUADRATIC. */
IF QUAD(1)=0 THEN DO;
IF QUAD(2)=0 THEN GO TO NONE;
IF MOD(QUAD(3),QUAD(2))~=0 THEN GO TO NONE;
K1 = QUAD(3)/QUAD(2);
GO TO L2;
END;
DISC = (QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
IF DISC < 0 THEN GO TO NONE;
IF MOD(SORT(DISC),1)~=0 THEN GO TO NONE;
TERM = SORT(DISC);
KK1 = -QUAD(2)-TERM;
IF MOD(KK1,2*QUAD(1))~=0 THEN GO TO NEXT1;
K1 = KK1/(2*QUAD(1));
L2: IF MOD(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE),COEFFS(2,1))~=0 THEN
GO TO NEXT1;
A1=(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE))/COEFFS(2,1);
FLAG=3;
GO TO GOTONE;

/* TRY OTHER SOLUTION OF QUADRATIC. */
NEXT1: IF COEFFS(2,1) = 0 THEN DO; A1 = -DISC;
FLAG=4; GO TO GOTONE; END;

IF QUAD(1) = 0 THEN GO TO NONE;
KK1=-QUAD(2)+TERM;
IF MOD(KK1,2*QUAD(1))~=0 THEN GO TO NONE;
K1 = KK1 / (2*QUAD(1));
IF MOD(COEFFS(1,1),(DEGREE*KK1)+(COEFFS(2,1))~=0 THEN GO TO NONE;
A1=COEFFS(1,1)/((DEGREE*KK1)+(COEFFS(2,1)));
FLAG = 4;
GO TO GOTONE;

```

TES02410
TES02420
TES02430
TES02440
TES02450
TES02460
TES02470
TES02480
TES02490
TES02500
TES02510
TES02520
TES02530
TES02540
TES02550
TES02560
TES02570
TES02580
TES02590
TES02600
TES02610
TES02620
TES02630
TES02640
TES02650
TES02660
TES02670
TES02680
TES02690
TES02700
TES02710
TES02720
TES02730
TES02740
TES02750
TES02760
TES02770
TES02780
TES02790
TES02800
TES02810
TES02820
TES02830
TES02840
TES02850
TES02860
TES02870
TES02880


```

/* IF NO TRANSFORMATIONS, INFORM OPERATOR. */
NONE:DISPLAY('NO TRANSFORMATIONS');
IF CHOICE='Y', THEN GO TO SECOND;
ELSE GO TO START;

/* IF POSSIBLE VALUE FOR K AND A FOUND, THEN CALCULATE TRIAL
POLYNOMIAL FOR COMPARISON. */

GOTONE:CALL TRANSFO;

/* PROCESS RESULT OF COMPARISON. */

/* IF COMPARISON CHECKS, THEN PRINT RESULT AND K AND A. */
IF ANSWER='YES', THEN DO;
PUT STRING(TEMP) EDIT ('K=',K1,'...A=',A1)(A,F(8),A,F(8));
TEMP1='';
DO K=1 TO LENGTH(TEMP);
IF SUBSTR(TEMP,K,1)=-, THEN TEMP1=TEMP1||SUBSTR(TEMP,
K,1);
END;
CHECK=1 THEN DISPLAY(TEMP1);
ELSE DISPLAY(TEMP1||'***');

/* RETURN FOR NEXT POLYNOMIAL. */
IF CHOICE='Y', THEN GO TO SECOND;
ELSE GO TO START;
END;

/* IF COMPARISON FAILS, THEN RETURN TO NEXT POINT OF CALCULATION. */
ELSE DO; IF FLAG = 1 THEN GO TO NEXT;
IF FLAG = 2 THEN GO TO OTHER;
IF FLAG = 3 THEN GO TO NEXT1;
IF FLAG = 4 THEN GO TO NONE;
END;

DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));
FACT:PROCEDURE (N) FIXED BIN (31) RECURSIVE;
/* FACT COMPUTES N FACTORIAL */
DCL (F,N) FIXED BIN (31);
IF N ^= 0 THEN GO TO SKIP;

```

TES02890
TES02900
TES02910
TES02920
TES02930
TES02940
TES02950
TES02960
TES02970
TES02980
TES02990
TES03000
TES03010
TES03020
TES03030
TES03040
TES03050
TES03060
TES03070
TES03080
TES03090
TES03100
TES03110
TES03120
TES03130
TES03140
TES03150
TES03160
TES03170
TES03180
TES03190
TES03200
TES03210
TES03220
TES03230
TES03240
TES03250
TES03260
TES03270
TES03280
TES03290
TES03300
TES03310
TES03320
TES03330
TES03340
TES03350
TES03360


```

F = 1;
RETURN (F);
SKIP:N=N-1;
F = FACT(N);
N = N+1;
F = F*N;
RETURN (F);
END FACT;

TRANSFO:PROCEDURE;
/*TRANSFO PERFORMS THE TRIAL TRANSFORMATIONS AND TESTS THE RESULT
AGAINST THE OTHER POLYNOMIAL */
DCL (J,K,L,M,TERM3) FIXED BIN(31);
DCL (BASE(0:DEGREE), TRIAL1(0:DEGREE), TRIAL2(0:DEGREE), TEST(0:DEGREE))
FIXED BIN(31);
IF A1 = 0 THEN DO;
ANSWER = 'NO';
RETURN; END;

IF FLAG = 1 | FLAG = 2 THEN DO; DO L = 0 TO DEGREE;
BASE(L)=COEFFS(1,L);
TEST(L)=COEFFS(2,L);
END;

IF FLAG = 3 | FLAG = 4 THEN DO; DO L = 0 TO DEGREE;
BASE(L)=COEFFS(2,L);
TEST(L)=COEFFS(1,L);
END; END;

/* PERFORM D TRANSFORMATION. */
DO K = 0 TO DEGREE;
M = ABS(A1)**K;
IF A1 < 0 THEN DO; IF MOD(K,2)=1 THEN M = -M; END;
TRIAL1(K) = BASE(K)*M;
END;

/* PERFORM T TRANSFORMATION. */
DO J = 0 TO DEGREE;
TERM3 = 0;
DO I = 0 TO J;
IF K1 /= 0 THEN M=ABS(K1)**(J-I);
ELSE DO; TRIAL2(J) = TRIAL1(J);
GO TO TAB; END;
IF K1 < 0 THEN DO; IF MOD(J-I,2)=1 THEN M=-M; END;
TERM3 = TERM3 + (TRIAL1(I)*FACT(DEGREE-I)*M)/FACT(J-I);
END;

```

TES033370
TES033380
TES033390
TES033400
TES033410
TES033420
TES033430
TES033440
TES033450
TES033460
TES033470
TES033480
TES033490
TES033500
TES033510
TES033520
TES033530
TES033540
TES033550
TES033560
TES033570
TES033580
TES033590
TES033600
TES033610
TES033620
TES033630
TES033640
TES033650
TES033660
TES033670
TES033680
TES033690
TES033700
TES033710
TES033720
TES033730
TES033740
TES033750
TES033760
TES033770
TES033780
TES033790
TES033800
TES033810
TES033820
TES033830
TES033840


```

TRIAL2(J) = TERM3/FACT(DEGREE-J);
TAB:END;
ANSWER='YES';

/* TEST RESULT. */
DO J = 0 TO DEGREE;
IF TRIAL2(J)≠TEST(J) THEN DO; ANSWER='NO'; RETURN; END;
END;

END TRANSFO;

MONIC:PROCEDURE;

/* MONIC TRANSFORMS A GIVEN POLYNOMIAL INTO A MONIC VIA
THE M TRANSFORMATION. */
DCL (M,K,L,AMOUNT) FIXED BIN (31);
DCL HOLD(0:DEGREE) FIXED BIN (31);
AMOUNT = COEFS(FLAG,0);
HOLD(0)=1;
HOLD(1)=COEFS(FLAG,1);
DO K = 2 TO DEGREE;
M=ABS(AMOUNT)**(K-1);
IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M = -M;
HOLD(K)= COEFS(FLAG,K)*M;
END;
TEMP = '';
DO K = 0 TO DEGREE;
COEFS(FLAG,K) = HOLD(K);
PUT STRING(RESPONSE) EDIT(HOLD(K))(F(15));
NUMBER = '';
DO L = 1 TO LENGTH(RESPONSE);
IF SUBSTR(RESPONSE,L,1)≠' ' THEN NUMBER=NUMBER||SUBSTR(RESPONSE,
L,1);
END;
TEMP = TEMP||NUMBER||',';
END;
TEMP = SUBSTR(TEMP,1,LENGTH(TEMP)-1);
PUT STRING(RESPONSE) EDIT('MONIC FOR POLY NUMBER',FLAG,' IS:')
(A,F(1),A);
DISPLAY(RESPONSE);
DISPLAY(TEMP);

END MONIC;

END; /* END OF BEGIN BLOCK */

```

TES03850
 TES03860
 TES03870
 TES03880
 TES03890
 TES03900
 TES03910
 TES03920
 TES03930
 TES03940
 TES03950
 TES03960
 TES03970
 TES03980
 TES03990
 TES04000
 TES04010
 TES04020
 TES04030
 TES04040
 TES04050
 TES04060
 TES04070
 TES04080
 TES04090
 TES04100
 TES04110
 TES04120
 TES04130
 TES04140
 TES04150
 TES04160
 TES04170
 TES04180
 TES04190
 TES04200
 TES04210
 TES04220
 TES04230
 TES04240
 TES04250
 TES04260
 TES04270
 TES04280
 TES04290
 TES04300
 TES04310
 TES04320

FINISH:DISPLAY('END OF JOB');
END TESTERA;

TES04330
TES04340


```

LUT2:PROC OPTIONS (MAIN);

/* LUT2 IS A PROGRAM WHICH ACCEPTS A SET OF FUNCTIONAL
   VALUES AND CALCULATES THE LAGRANGIAN POLYNOMIAL
   THEM AND TESTS IT FOR MONICITY AND INTEGRAL
   COEFFICIENTS. */

ON ENDFILE (SYSIN) GO TO FINIS;
DCL DEG FLOAT BIN (53);
DCL (X,Y) FLOAT BIN (53) INITIAL (0);

/* DETERMINE DEGREE. */

GET LIST (DEG);
BEGIN;

/* SET UP NECESSARY ARRAYS FOR DEGREE SPECIFIED. */

DCL FVAL(0:DEG) FLOAT BIN (53);
DCL MPWIC(0:DEG) FLOAT BIN (53);

/* GET SET OF FUNCTIONAL VALUES. */

START:DO K = 0 TO DEG;
  GET LIST (FVAL(K));
END;

/* CALL PROCEDURE TO CALCULATE LAGRANGIAN POLYNOMIAL. */

CALL LAGRANGE(DEG, FVAL);

/* TEST FOR MONICITY/ */

IF ABS(MPWIC(0)) = 1 THEN DO; CALL PRINTER;
  Y = Y+1;
END;

/* INCREMENT COUNTER. */

X = X+1;

/*RETURN FOR NEXT SET OF VALUES. */

GO TO START;

LAGRANGE:PROCEDURE (N,ABCISSAE);

```



```

/*LAGRANGE IS A PROCEDURE WHICH CALCULATES THE
POLYNOMIAL AND TESTS IT FOR INTEGRAL COEFFICIENTS. */

      DCL (J,K,L,M,N,DENOM) FLOAT BIN(53);
      DCL (ABCISSAE(:N),TERMS(O:N,O:N), ORDINATES(O:N),
      FACTORS(N), POLY(O:N), DENOMS(O:N)) FLOAT BIN(53);

/* LOAD ARRAYS. */
      DO K = 0 TO N;
        MPWIC(K) = 0;
        ORDINATES(K) = K;
      END;

/* CALCULATE POLYNOMIAL. */
      DO J = 0 TO N;
        DENOM = 1;
        DO L = 0 TO J-1, J+1 TO N;
          DENOM = DENOM * (ORDINATES(J) - ORDINATES(L));
        END; DENOMS(J) = DENOM;

        M = 0;
        DO L = 0 TO J-1, J+1 TO N;
          M = M + 1;
          FACTORS(M) = (-ORDINATES(L));
        END;
        POLY(N) = FACTORS(N);
        POLY(N-1) = 1;
        DO L = N-2 TO 0 BY -1;
          POLY(L) = 1;
          DO K = 1 TO (N-L-1);
            POLY(L+K) = POLY(L+K+1) + FACTORS(L+1)*POLY(L+K);
          END;
          POLY(N) = POLY(N)*FACTORS(L+1);
        END;
        POLY = POLY * ABCISSAE(J);
        TERMS(J,*) = POLY;
      END;
      DO L = 0 TO N;
        DO K = 0 TO L-1, L+1 TO N;
          TERMS(L,*) = TERMS(L,*)*DENOMS(K);
        END;
      END;
      DO L = 0 TO N;

```



```

DO K = 1 TO N;
  TERMS(O,L) = TERMS(O,L)+TERMS(K,L);
END;
DO L = 1 TO N;
  DENOMS(O) = DENOMS(O) * DENOMS(L);
END;
DO L = 0 TO N;
  IF MOD(TERMS(O,L), DENOMS(O)) = 0 THEN
    TERMS(O,L) = TERMS(O,L)/DENOMS(O);
  ELSE DO; DO K = 0 TO N;
    MPWIC(K)=0;
  END;
  RETURN;
END;
END LAGRANGE;

PRINTER:PROCEDURE;
/* PRINTER PRINTS THE POLYNOMIAL OR, IF THE FIRST
   COEFFICIENT IS -1, IT PRINTS THE REFLECTION. */
IF MPWIC(O) = 1 THEN DO; PUT EDIT('X',DEG)(SKIP,A,F(1,0));
DO P = 1 TO DEG-2;
  PUT EDIT(' ',MPWIC(P),' X',DEG-P)
    (A,F(4,0),A,F(1,0));
END;
PUT EDIT(' ',MPWIC(DEG-1),' X ',
  MPWIC(DEG))(A,F(4,0),A,F(4,0));
DO P = 0 TO DEG;
  PUT EDIT(' ',FVAL(P))(A,F(2));
END;
ELSE DO; PUT EDIT('X',DEG)(SKIP,A,F(1,0));
DO P = 1 TO DEG-2;
  PUT EDIT(' ', -MPWIC(P),' X',DEG-P)
    (A,F(4,0),A,F(1,0));
END;
PUT EDIT(' ', -MPWIC(DEG-1),' X ',
  -MPWIC(DEG))(A,F(4,0),A,F(4,0));
DO P = 0 TO DEG;
  PUT EDIT(' ', -FVAL(P))(A,F(2));
END;
END PRINTER;

```



```
END;      /* END OF BEGIN BLOCK. */  
FINIS:PUT EDIT('CANDIDATES=',X,'  
END LUTZ2;      MPWICS=',Y)(SKIP,A,F(4),A,F(4));
```



```

TESTER:PROCEDURE OPTIONS(MAIN);

/* TESTER IS A PROGRAM WHICH SEPARATES A DATA SET OF MIPWIC'S
   INTO EQUIVALENCES CLASSES. */
/* DOCUMENTATION IS SAME AS FOR TESTERA EXCEPT
   AS NOTED. */

CN ENDFILE (SYSIN) GO TO FINISH;
DCL RESPONSE CHAR(512) VARYING INITIAL ('');
DCL BUFFER CHAR(80);
DCL DEGREE FIXED BIN (31);
DCL (TEMP, TOP1) CHAR(72) VARYING;
DCL NUMBER CHAR(15) VARYING;
DCL ANSWER CHAR(15) VARYING;
DCL (KK1, K1) FIXED BIN(31);
DCL DIGITS CHAR(10) INITIAL ('0123456789');
DCL DIGITS1 CHAR(11) INITIAL ('0123456789-');
DCL STRING1 CHAR(12) INITIAL ('0123456789-');
DCL (L, J, K, FLAG) FIXED BIN (31);
DCL (A1, A2) FIXED BIN (31);
DCL DISC FIXED BIN (31, 5);
DCL TERM FIXED BIN (31);
DCL CHECK FIXED BIN (15);
DCL ERROR CHAR(25) INITIAL ('INVALID RESPONSE--REENTER');
DCL CHOICE CHAR(4) VARYING INITIAL ('');
DCL (SET, INDEX1, INDEX2) FIXED BIN (31);

/* GET DEGREE AND NUMBER OF POLYNOMIALS IN DATA SET. */
START:GET LIST (DEGREE, SET);
BEGIN;
/* DECLARE ARRAYS TO PROPER SIZE. */
DCL COEFFS(2, 0:DEGREE) FIXED BIN (31);
DCL QUAD(3) FIXED BIN (31);
DCL DATA(SET, 0:DEGREE+1) FIXED BIN (31);
/* GET FIRST POLYNOMIAL FOR TESTING. */
DO K = 1 TO SET;
  DO J = 0 TO DEGREE;
    GET LIST (DATA(K, J));
  END;
  DATA(K, DEGREE+1)=1;
END;
/* FIND NEXT POLYNOMIAL FOR TESTING AGAINST FIRST. */
INDEX1=0;
TOP1:INDEX1=INDEX1+1;
IF INDEX1=SET+1 THEN GO TO START;
IF DATA(INDEX1, DEGREE+1)=0 THEN GO TO TOP1;
DO K = 0 TO DEGREE;

```



```

COEFFS(1,K)=DATA(INDEX1,K);
END;
PUT EDIT('***')(SKIP,A);
RESPONSE='';
DO K=0 TO DEGREE;
  PUT STRING(TEMP1) EDIT(COEFFS(1,K))(F(8));
  RESPONSE=RESPONSE||TEMP1;
END;
DO K=2 TO LENGTH(RESPONSE);
  IF SUBSTR(RESPONSE,K,1)='1' THEN RESPONSE=SUBSTR(RESPONSE,
1,K-1)||SUBSTR(RESPONSE,K+1);
END;
SUBSTR(RESPONSE,1,1)='1' THEN RESPONSE=SUBSTR(RESPONSE,2);
PUT EDIT(RESPONSE)($KIP,A);
INDEX2=INDEX1;
IF INDEX2=INDEX2+1; THEN GO TO TOP1;
IF DATA(INDEX2,DEGREE+1)=0 THEN GO TO TOP2;
DO K=0 TO DEGREE;
  COEFFS(2,K)=DATA(INDEX2,K);
END;
IF COEFFS(1,0)=-1 THEN DO;FLAG=1;CALL MONIC; END;
IF COEFFS(2,0)=-1 THEN DO;FLAG=2;CALL MONIC; END;
IF COEFFS(1,1)=0 & COEFFS(2,1)=0 THEN DO;
  IF MOD(COEFFS(2,DEGREE)/COEFFS(1,DEGREE))=0 THEN DO;
    KK1=COEFFS(2,DEGREE)/COEFFS(1,DEGREE);
    L=0;
  TAB1:L=L+1;
  TERM=1;
  DO J=1 TO DEGREE;
    TERM=TERM*L;
  END;
  IF TERM < KK1 THEN GO TO TAB1;
  IF TERM = KK1 THEN DO;
    A1=L;
    K1=0;
    FLAG=2;
    CHECK=1;
    GO TO GOTONE;
  END;
END;
IF MOD(COEFFS(1,DEGREE)/COEFFS(2,DEGREE))=0 THEN DO;
  KK1=COEFFS(1,DEGREE)/COEFFS(2,DEGREE);
  L=0;
  TAB2:L=L+1;
  TERM=1;
  DO J=1 TO DEGREE;
    TERM=TERM*L;

```



```

END;
IF TERM < KK1 THEN GO TO TAB2;
IF TERM = KK1 THEN DO;
  A1=L;
  K1=0;
  FLAG=4;
  CHECK=0;
  GO TO GOTONE;
END;
END;
GO TO TOP2;
END;
CHECK = 1;
IF COEFFS(1,1)≠0 THEN DO;
  IF MOD(COEFFS(2,1),(COEFFS(1,0)*DEGREE))≠0 THEN GO TO OTHER;
  K1=COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
  DISC=COEFFS(1,0)*DEGREE*(COEFFS(2,2)
    -((COEFFS(2,1)**2)*(DEGREE-1))/
    (2*COEFFS(1,2)*COEFFS(1,0)*DEGREE) );
  IF DISC < 0 THEN GO TO OTHER;
  DISC=SQRT(DISC);
  IF MOD(DISC,1)≠0 THEN GO TO OTHER;
  A1=DISC;
  FLAG=1;
  GO TO GOTONE;
END;
QUAD(1)=(COEFFS(1,0)*DEGREE*(COEFFS(1,1)**2)*(1-DEGREE))+
  (2*(COEFFS(1,0)**2)*(DEGREE**2)*COEFFS(1,2));
QUAD(2)=(2*(COEFFS(2,1)*DEGREE*(COEFFS(1,1)**2)-(2*COEFFS(1,2)
  *COEFFS(1,0))-(2*COEFFS(2,1)*(COEFFS(1,1)**2)));
QUAD(3)=(2*(COEFFS(1,2)*(COEFFS(2,1)**2)-(2*(COEFFS(1,1)**2)*
  COEFFS(2,2)));
IF QUAD(1)=0 THEN DO;
  IF QUAD(2)=0 THEN GO TO OTHER;
  IF MOD(QUAD(3),QUAD(2))≠0 THEN GO TO OTHER;
  K1=QUAD(3)/QUAD(2);
  GO TO L1;
END;
DISC=(QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
IF DISC < 0 THEN GO TO OTHER;
IF MOD(SQRT(DISC),1)≠0 THEN GO TO OTHER;
TERM = SQRT(DISC);
KK1 = -QUAD(2)-TERM;
IF MOD(KK1,2*QUAD(1))≠0 THEN GO TO NEXT;
K1=KK1/(2*QUAD(1));
L1: IF MOD(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE),COEFFS(1,1))≠0
  THEN GO TO NEXT;
  A1=(COEFFS(2,1)-(COEFFS(1,0)*K1*DEGREE))/COEFFS(1,1);

```



```

FLAG=1;
GO TO GOTONE;
NEXT:IF COEFFS(1,1)=0 THEN DO: A1=-DISC;
FLAG=2;
GO TO GOTONE;

IF QUAD(1)=0 THEN GO TO OTHER;
KK1=-QUAD(2)+TERM;
IF MOD(KK1,2*QUAD(1))=0 THEN GO TO OTHER;
K1=KK1/(2*QUAD(1));
IF MOD(COEFFS(2,1)-(DEGREE*K1)+(COEFFS(1,1)))=0 THEN GO TO OTHER;
A1=COEFFS(2,1)/(DEGREE*K1+(COEFFS(1,1)));
FLAG=2;
GO TO GOTONE;
OTHER:CHECK=C;
IF COEFFS(2,1)=0 THEN DO:
  IF MOD(COEFFS(1,1),(COEFFS(2,0)*DEGREE))=0 THEN GO TO NONE;
  K1=COEFFS(2,1)/(COEFFS(1,0)*DEGREE);
  DISC=(2*COEFFS(2,0)*DEGREE*COEFFS(1,2))
    -((COEFFS(1,1)**2)*(DEGREE-1))/
    (2*COEFFS(2,2)*COEFFS(2,0)*DEGREE);
  IF DISC < 0 THEN GO TO NONE;
  DISC = SQRT(DISC);
  IF MOD(DISC,1)=0 THEN GO TO NONE;
  A1=DISC;
  FLAG=3;
  GO TO GOTONE;
END;
QUAD(1)=(COEFFS(2,0)*DEGREE*(COEFFS(2,1)**2)*(1-DEGREE))
+ (2*(COEFFS(2,0)**2)*(DEGREE**2)*COEFFS(2,2));
QUAD(2)=(2*COEFFS(1,1)*DEGREE)*(COEFFS(2,1)**2)-(2*COEFFS(2,2)
*COEFFS(2,0))-(2*COEFFS(1,1)*(COEFFS(2,1)**2));
QUAD(3)=(2*COEFFS(2,2)*(COEFFS(1,1)**2)-(2*(COEFFS(2,1)**2)
*COEFFS(1,2));
IF QUAD(1)=0 THEN DO:
  IF QUAD(2)=0 THEN GO TO NONE;
  IF MOD(QUAD(3),QUAD(2))=0 THEN GO TO NONE;
  K1=QUAD(3)/QUAD(2);
  GO TO L2;
END;
DISC = (QUAD(2)**2)-(4*QUAD(1)*QUAD(3));
IF DISC < 0 THEN GO TO NONE;
IF MOD(SQRT(DISC),1)=0 THEN GO TO NONE;
TERM = SQRT(DISC);
KK1 = -QUAD(2)-TERM;
IF MOD(KK1,2*QUAD(1))=0 THEN GO TO NEXT1;
K1=KK1/(2*QUAD(1));
L2: IF MOD(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE),COEFFS(2,1))=0 THEN

```



```

GO TO NEXT1;
A1=(COEFFS(1,1)-(COEFFS(2,0)*K1*DEGREE))/COEFFS(2,1);
FLAG=3;
GO TO GOTONE;
NEXT1: IF COEFFS(2,1)=0 THEN DO: A1=-DISC;
FLAG=4;
GO TO GOTONE;
END;
IF QUAD(1)=0 THEN GO TO NONE;
KK1=-QUAD(2)+TERM;
IF MOD(KK1,2*QUAD(1))=0 THEN GO TO NONE;
K1 = KK1 / (2*QUAD(1));
IF MOD(COEFFS(1,1),(DEGREE*K1)+(COEFFS(2,1)))=0 THEN GO TO NONE;
A1=COEFFS(1,1)/((DEGREE*K1)+(COEFFS(2,1)));
FLAG = 4;
GO TO GOTONE;
NONE: GO TO TOP2;
GOTONE: CALL TRANSFO;
IF ANSWER='YES' THEN DO: RESPONSE = '';
DO K = 0 TO DEGREE;
PUT STRING (TEMP) EDIT (COEFFS(2,K))(F(8));
RESPONSE = RESPONSE||TEMP;
END;
DO K = 2 TO LENGTH(RESPONSE);
IF SUBSTR(RESPONSE,K,1) = ' ' THEN RESPONSE =
SUBSTR(RESPONSE,1,K-1)||SUBSTR(RESPONSE,K+1);
END;
IF SUBSTR(RESPONSE,1,1) = ' ' THEN RESPONSE=SUBSTR(RESPONSE,
2);
PUT STRING (TEMP) EDIT (RESPONSE||' K=' ,K1,' A=' ,A1)
(A,F(5),A,F(5));
IF CHECK = 1 THEN PUT EDIT (TEMP) (SKIP,A);
ELSE DO: TEMP=TEMP||'***';
PUT EDIT (TEMP) (SKIP,A);
END;
DATA(INDEX2,DEGREE+1)=0;
GO TO TOP2;
END;
ELSE DO: IF FLAG=1 THEN GO TO NEXT;
IF FLAG = 2 THEN GO TO OTHER;
IF FLAG = 3 THEN GO TO NEXT1;
IF FLAG = 4 THEN GO TO NONE;
END;
DCL FACT ENTRY (FIXED BIN (31)) RETURNS (FIXED BIN (31));

FACT: PROCEDURE (N) FIXED BIN (31) RECURSIVE;
DCL (F,N) FIXED BIN (31);

```



```

IF N /= 0 THEN GO TO SKIP;
F = 1;
RETURN (F);
SKIP: N=N-1;
F = FACT(N);
N = N+1;
F = F*N;
RETURN (F);
END FACT;

TRANSFO: PROCEDURE;
DECL (J,K,L,M,TERM3) FIXED BIN(31);
DECL (BASE(O:DEGREE), TRIAL1(O:DEGREE), TRIAL2(O:DEGREE), TEST(O:DEGREE))
FIXED BIN(31);
IF A1=0 THEN DO;
ANSWER='NO';
RETURN;
END;
IF FLAG = 1 | FLAG = 2 THEN DO; DO L = 0 TO DEGREE;
BASE(L)=COEFFS(1,L);
TEST(L)=COEFFS(2,L);
END;
IF FLAG = 3 | FLAG = 4 THEN DO; DO L = 0 TO DEGREE;
BASE(L)=COEFFS(2,L);
TEST(L)=COEFFS(1,L);
END;
DO K = 0 TO DEGREE;
M = ABS(A1)**K;
IF A1 < 0 THEN DO; IF MOD(K,2)=1 THEN M = -M; END;
TRIAL1(K) = BASE(K)*M;
END;
DO J = 0 TO DEGREE;
TERM3 = C;
DO I = 0 TO J;
IF K1 /= 0 THEN M=ABS(K1)**(J-I);
TRIAL2(J) = TRIAL1(J);
ELSE DO; GO TO TAB; END;
IF K1 < 0 THEN DO; IF MOD(J-I,2)=1 THEN M=-M; END;
TERM3 = TERM3 + (TRIAL1(I)*FACT(DEGREE-I)*M)/FACT(J-I);
END;
TRIAL2(J) = TERM3/FACT(DEGREE-J);
TAB: END;
ANSWER='YES';
DO J = 0 TO DEGREE;
IF TRIAL2(J) /= TEST(J) THEN DO; ANSWER='NO'; RETURN; END;
END TRANSFO;

```



```

MONIC:PROCEDURE;
DCL (M,K,L,AMOUNT) FIXED BIN (31);
DCL HOLD(0:DEGREE) FIXED BIN (31);
AMOUNT = COEFFS(FLAG,0);
HOLD(0)=1;COEFFS(FLAG,1);
HOLD(1)=COEFFS(FLAG,1);
DO K = 2 TO DEGREE;
M=ABS(AMOUNT)**(K-1);
IF AMOUNT < 0 & MOD(K-1,2)=1 THEN M = -M;
COEFFS(FLAG,K) = COEFFS(FLAG,K) * M;
END;
END MONIC;

END;
FINISH:PUT EDIT ('END OF JOB') (SKIP,A);
END TESTER;

```


BIBLIOGRAPHY

1. Dorwart, H. L., "Irreducibility of Polynomials," The American Mathematical Monthly, V. XLII, p. 369-381, June-July 1935.
2. Kelly, J. B., "On Factorization of Polynomials," The American Mathematical Monthly, V. 60, p. 375-379, June-July 1953.
3. Knuth, D. E., The Art of Computer Programming, V. 2, p. 360-444, Addison-Wesley, 1969.
4. Lehmer, D. H., "Computer Technology Applied to the Theory of Numbers," Studies in Mathematics, V. 6, p. 117-151, The Mathematical Association of America, 1969.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Asst Professor Daniel L. Davis, Code 53Dv Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4. LT Robert E. Lutz, USN 409 East Water Street Troy, Ohio 45373	1

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

UNCLASSIFIED

2b. GROUP

3. REPORT TITLE

AN IMPLEMENTED TRANSFORMATIONAL SCHEME FOR MONIC POLYNOMIALS
WITH INTEGER COEFFICIENTS

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis, June 1971

5. AUTHOR(S) (First name, middle initial, last name)

Robert E. Lutz, Jr.

6. REPORT DATE

June 1971

7a. TOTAL NO. OF PAGES

82

7b. NO. OF REFS

4

8a. CONTRACT OR GRANT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

b. PROJECT NO.

c.

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned
this report)

d.

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A discussion of the problem of the irreducibility of polynomials in the ring of integral polynomials establishes the framework of the research. A transformational scheme is postulated to facilitate investigation of the problem. The coherency of the scheme is detailed and the necessary computational techniques developed.

To determine the efficacy of the transformational scheme, the specification and collection of appropriate sets of data are discussed. The transformational scheme is then applied to the data and the results are tabulated and discussed.

The conclusion is drawn that a transformational scheme is a useful tool for the investigation of the irreducibility of this type of polynomial and some hypotheses as to the refinement and extension of the technique are stated.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Irreducible Polynomials						
Computer-Implemented Transformational Scheme						

Thesis

L93

c.1

Lutz

An implemented transformational scheme for monic polynomials with integer coefficients.

128470

Thesis

L93

c.1

Lutz

An implemented transformational scheme for monic polynomials with interger coefficients.

128470

thesL93

An implemented transformational scheme f



3 2768 002 12423 2

DUDLEY KNOX LIBRARY